ECOLE NATIONALE SUPERIEURE DE L'AERONAUTIQUE
DIRECTION DES STAGES DE PERFECTIONNEMENT

ASSERVISSEMENTS ET AUTOMATISMES

" PROGRAMMATION DE L'ALGORITHME DE SYNTHESE DE GLOUSHKOV POUR LES TABLES DE FLUENCE DES SYSTEMES SEQUENTIELS "

Dans le cadre d'un programme
d'obtention du diplôme de
Maître ès-Sciences Aéronautiques

Fernando Sáez Vacas
Stages Postscolaires
Août 1966

Je tiens à remercier:

La Direction des Stages de l'E.N.S.A,

Monsieur E.Daclin qui a accepté de diriger ce travail,

Monsieur J.P.Perrin dont les conseils m'ont été très profitables,

Madame Renault qui m'a aidé beaucoup au pied de la machine CAB-500.

Monsieur M. Pélégrin qui a mis à ma disposition les moyens téchniques du C.E.R.A.

S O M M A I R E

A l'origine nous nous avions proposé programmer sur CAB-500 la méthode de synthèse de Gloushkov pour la classe des automates asynchrones du type machine transfert, d'où le titre de la thèse.

A sa place nous avons resous le même problème à l'aide d'une méthode algorihmique originale.

Après une introduction on définit la nouvelle méthode, valable pour les machines asynchrones, ainsi que quelques propriétés intéréssantes des expressions itérées (en particulier de l'événement universel). Dans la suite on établit les organigrammes générales de synthèse et l'organisation du travail sur machine.

Après la conclusion, où l'on résume les avantages de nôtre méthode, il y a trois annexes, dans le premier desquels on fait d'une façon pratique le point sur ces avantages par rapport à la méthode de Gloushkov , dans le deuxième on groupe des organigrammes très detaillés, les programmes correspondants et quelques résultats , et dans le troisième le programme traduit en fortran IV qui a été mis au point sur le calculateur IBM 360/44 du C.E.R.A. (Centre d'Etudes et Récherches en Automatisme).

T A B L E D E M A T I E R E S

C H A P I T R E I

INTODUCTION DU PROBLEME

1.1.- Généralités

1.11.- La synthèse des tables

Lors de la conception d'une machine sequentielle on connaît une correspondance entre certaines séquences d'entrée et cetaines séquences de sortie. Il faut, à partir de là, trouver la table de fluence qui amènera ensuite à la table d'excitation et à la table de sortie. Ceci est la synthèse des tables, qui est normalement réalisée en deux stades bien differenciés: passage à la table de fluence et réduction des états internes.

On peut donner le schéma général suivant (figure 1.1) du chemin suivi dans la synthèse d'un système sequentiel:

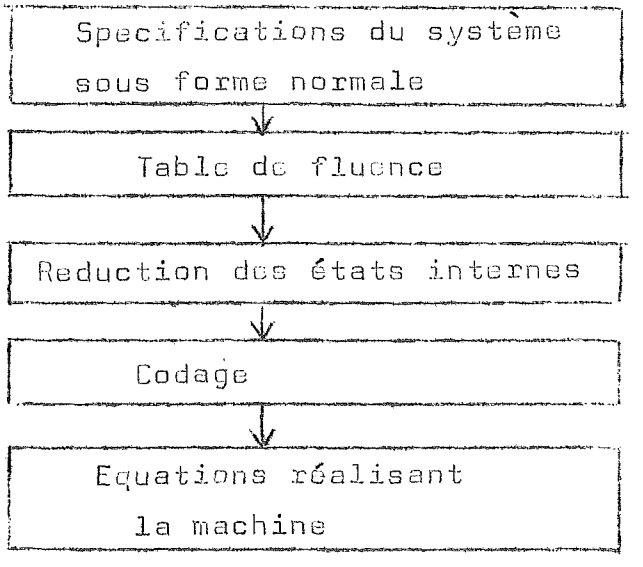


FIG. 1.1

Pour faire le premier passage (données ⟹ table de fluence) une méthode commode consiste à employer les expressions regulières suivant un algorithme dû à Gloushkov. L'efficacité de cette méthode (lorsqu'on la considère au point de vue de la vitesse du travail et de la diminution du risque d'erreur) doît être considerablement augmentée par l'utilisation d'une machine numérique.

Le but de notre thèse est d'automatiser ce passage pour un certain type d'expression regulière que nous preciserons plus loin. La machine numérique utilisée est la CAB-500.

1.12.- Apercu sur les diverses méthodes de synthèse

Depuis quelques années (ref. 1) une partie très importante de l'effort dans la théorie des automates finis a été dirigée vers le developpement des algorithmes qui peuvent être programmés sur calculateur digital.

Au debut ce furent seulement des machines speciales conçues surtout pour l'analyse des systèmes logiques. Le succès de ces machines stimula la recherche des programmes sur calculateur numérique en vue de la conception des circuits combinatoires, et plus tard des circuits séquentiels. On comprend bien que cette tendance a été dans une certaine mesure preparée et favorisée par plusieurs facteurs, les uns attachés aux autres: d'abord la complexité croissante des problèmes à résoudre, puis le perfectionnement de la technologie et la diminution du coût des composants, ce qui amène à la construction pratique des réseaux beaucoup plus importants, donc à des calculateurs de plus en plus puissants et rapides.

Ceci veut dire que le théoricien s'est vu contraint

à chercher des algorithmes efficaces, des procédés systématiques qui permettent de laisser à une machine numérique le lourd volume de travail du traitement d'un système logique.

Parmi les algorithmes proposés (ref. 2) pour le premier stade de la synthèse des tables, ceux dûs à Ginsburg, Aizerman, Moisil-Ioanin et Gloushkov, ce dernier présente un avantage considérable par rapport aux trois autres, c'est qu'il est aussi bien valable pour les systèmes synchrones que pour les asynchrones. Au contraire les deux premiers ne sont valables que pour les machines synchrones et la méthode de Moisil seulement pour les asynchrones.

Au point de vue programmation la méthode de Gloushkov est aussi plus avantageuse que celles de Gisburg et Moisil puisqu'elle amène à un encombrement moins important des mémoires. La méthode d'Aizerman serait meilleure à ce point de vue si elle pouvait permettre de réaliser des machines correspondant à des évènements du type $R = \{ i \} P$, très fréquents dans la technique, ainsi que les évènements conduisant à des machines asynchrones.

Toutes ces méthodes ont quand même deux choses en commun, en ce sens qu'elles deviennent impraticables traitées à la main dès que le problème est d'ordre industriel (à partir de quinze variables), d'où la nécessité de leur automatisation par calculateur et qu'elles supposent que les spécifications sont exprimées sous la forme normale.

1.2.- Spécifications du problème sous forme normale

On peut écrire les équations générales d'un système séquentiel sous la forme:

$$\begin{cases} Z_n = F(X_n, Q_n) \\ Q_{n+1} = G(X_n, Q_n) \end{cases}$$

où X_n, Z_n et Q_n représentent respectivement les vecteurs d'entrée, de sortie et d'état interne à l'instant n.

La première étape de la synthèse consiste à passer d'un certain nombre de relations entre les alphabets d'entrée et de sortie (X) et (Z) à la détermination de (Q), F et G, obtenues souvent sous forme de tables.

On dira que ces relations de départ sont données sous forme normale lorsqu'elles sont représentées par des correspondances entre des séquences temporelles d'un certain nombre de vecteurs d'entrée et le même nombre de vecteurs de sortie.

Ainsi une correspondance entre (X) = (X_1, X_2, X_3) et (Z) = (Z_1, Z_2) peut être par exemple:

$X_2X_1X_3X_1X_2$ séquence d'entrée (instants 0, 1, 2, 3, 4 et 5)

$\Downarrow$

$Z_2Z_1Z_2Z_1Z_2$ séquence de sortie (instants 0, 1, 2, 3, 4 et 5)

Intéressons nous maintenant spécialement à un type particulier de séquence d'entrée: c'est la séquence itérée, qui consiste en l'application à la machine zéro fois, une fois,... m fois... d'une même suite d'entrées.

On note cette suite d'entrées entre deux accolades $\{\ \}$.

Ainsi (P) = $\{X_2X_1X_2\}$ signifie qu'on peut appliquer à la machine

- ou bien rien du tout (séquence de longueur nulle designée par λ)
- ou bien $X_2X_1X_2$
- ou bien $X_2X_1X_2 \quad X_2X_1X_2$
- ou bien $\underbrace{X_2X_1X_2 \quad X_2X_1X_2 \quad \ldots \quad X_2X_1X_2}_{\text{m fois}}$
- ou bien $X_2X_1X_2 \quad X_2X_1X_2 \ldots\ldots X_2X_1X_2 \quad \ldots$

L'énoncé du problème sous forme normale doit être donné de façon non contadictoire.

1.3.- Algèbre des expressions regulières

1.31.- Definitions et propriétés

Supposons une machine séquentielles, dont les alphabets d'entrée et de sortie sont ($X_1X_2X_3$) et (Z_1Z_2) et qui réalise la correspondance suivante entre des séquences d'entrées et de sorties:

$$X_1\ X_2\ X_1\ X_3\ X_1\ X_3\ X_2$$
$$\Downarrow$$
$$Z_1\ Z_2\ Z_2\ Z_1\ Z_2\ Z_2\ Z_1$$

Les expressions

$$R\ (\ Z_1\ ) = X_1 \vee X_1X_2X_1X_3 \vee X_1X_2X_1X_3X_1X_3X_2$$

$$R\ (\ Z_2\ ) = X_1X_2 \vee X_1X_2X_1 \vee X_1X_2X_1X_3X_1 \vee X_1X_2X_1X_3X_1X_3$$

(ou le symbole V signifie " ou bien ") expriment d'une façon séparée le fait de la réalisation de tous les vecteurs de l'alphabet de sortie pour plusieurs séquences possibles des vecteurs d'entrée.

Une expression du type R (Z) s'appelle expression regulière.

(Dans la suite nous remplacerons le signe V par un signe +, qui est plus maniable; on désignera toujours les variables d'entrée par des X indexées et d'autres lettres, telles que P, Q, etc..., réprésenteront des expressions regulières ou des parties des expressions regulières).

Ainsi on pourra écrire la première des relations ci-dessus de la façon suivante:

$$R\ (\ Z_1\ ) = P_1 + P_1P_2 + P_1P_2P_3$$

avec $P_1 = X_1$; $P_2 = X_2X_1X_3$; $P_3 = X_1X_3X_2$

Definition.- La séquence de longueur nulle Λ est celle qui

précédant ou suivant une séquence quelconque P ne modifie en rien celle-ci. Donc $P\Lambda = \Lambda P = P$.

Introduisons maintenant un nouvel ensemble

$$\{ i \} = \{ X_1 + X_2 + \ldots + X_N \}$$

qui, développé, devient:

$$\{ i \} = \Lambda + (X_1 + X_2 + \ldots + X_N) + (X_1 + X_2 + \ldots + X_N)^2 + \ldots + (X_1 + X_2 + \ldots + X_N)^m + \ldots =$$

$$= \sum_{m=0}^{\infty} (X_1 + X_2 + \ldots + X_N)^m$$

Or, $(X_1 + X_2 + \ldots + X_N)^m$ représente l'ensemble de toutes les séquences de longueur m qu'il est possible d'appliquer à l'aide de l'alphabet $(X) = (X_1 X_2 \ldots X_N)$. Par conséquent $\{ i \}$ represente l'ensemble des séquences de longueur quelconque que l'on peut appliquer à partir de l'alphabet d'entrée (X). On l'appelle évenement universel.

Il y a encore un autre ensemble $\emptyset$ des séquences indeterminées ou interdites qui nous intéresse moins. Mais ce qui est important c'est de résumer quelques propriétés de l'algèbre des expressions regulières dans le tableau suivant, emprunté textuellement à l'ouvrage cité en référence 2 (Perrin et al.).

$$P + Q = Q + P$$
$$P + (Q + S) = (P + Q) + S$$
$$P + P = P$$

$$P (QS) = (PQ) S$$
$$P (Q + S) = PQ + PS$$
$$(P + Q) S = PS + QS$$

$$\{\{P\}\} = \{P\}$$
$$\{P\} = \lambda + P\{P\}$$
$$P\{P\} = \{P\}P$$
$$\{P\}\{P\} = \{P\}$$
$$\{P\} + P = \{P\}$$

$$\lambda P = P\lambda = P \qquad \{\lambda\} = \lambda$$

$$P\emptyset = \emptyset P = \emptyset \qquad P + \emptyset = P \qquad \{\emptyset\} = \lambda$$

Toutes les règles d'indexation de Gloushkov découlent de ces propriétés-ci, ainsi qu'on verra dans un instant, à l'aide d'un exemple.

Au passage signalons que dans le chapitre 3 on va proposer une nouvelle notation pour les expressions itérées.

1.32.- Application au systèmes asynchrones

Jusqu'à maintenant on n'a rien dit mais on a sousentendu qu'il s'agissait toujours des variables d'entrée synchrones. Or, si l'on veut appliquer l'algèbre des expressions regulières au cas d'une machine asynchrone il faut d'abord trouver une façon valable de représentation des vecteurs asynchrones.

Rappelons qu'un système est dit asynchrone dans le cas où une modification des variables (X) d'entrée peut entraîner une ou une cascade de modifications transitoires qui, par passage par un ou plusieurs états instables intermédiaires, n'amèneront le système dans un nouvel état stable qu'après un temps essentiellement variable selon les modifications commandées (ref. 3).

En termes d'équation cela peut s'exprimer:

$$G\ (\ X_n,\ Q_n\ )\ =\ G\ (\ X_n X_n,\ Q_n\ )$$

Donc la diffrérence essentielle entre systèmes synchrones et asynchrones est liée à l'existence d'états stables dans ce deuxième type de systèmes.

La réprésentation d'une commande asynchrone par $C = X\{X\}$ doit être valable.

En effet,

$$C = X\{X\} = X + XX + XXX + \ldots + XX\ldots X + \ldots$$

ce qui veut dire que la commande peut être appliquée pendant Δ, 2Δ, 3Δ.... $m\Delta$, ... (Δ étant le temps de réaction du circuit): donc celui-ci peut evoluer de lui même.

Alors la séquence la plus générale serait

$$C = \ldots X_i \{ X_i \} X_j \ldots \quad (\text{ avec la condition } j \neq i\ )$$

où l'on peut démontrer très facilement par les règles d'extension d'indices de Gloushkov l'existence d'états stables dans la table de fluence.

1.4.- Synthèse des tables par la méthode de Gloushkov

Nous sommes maintenant en mesure, avant de conclure ce chapitre de préciser davantage et les lignes générales de la méthode de Gloushkov et le problème que nous allons envisager.

Reprenons le schéma de la figure 1.1 et y detaillons la manière dont Gloushkov résout la synthèse des tables à partir de la forme normale de l'énoncé (figure 1.2).

(Les operations encadrées sont celles qui correspondent au procédé de Gloushkov).

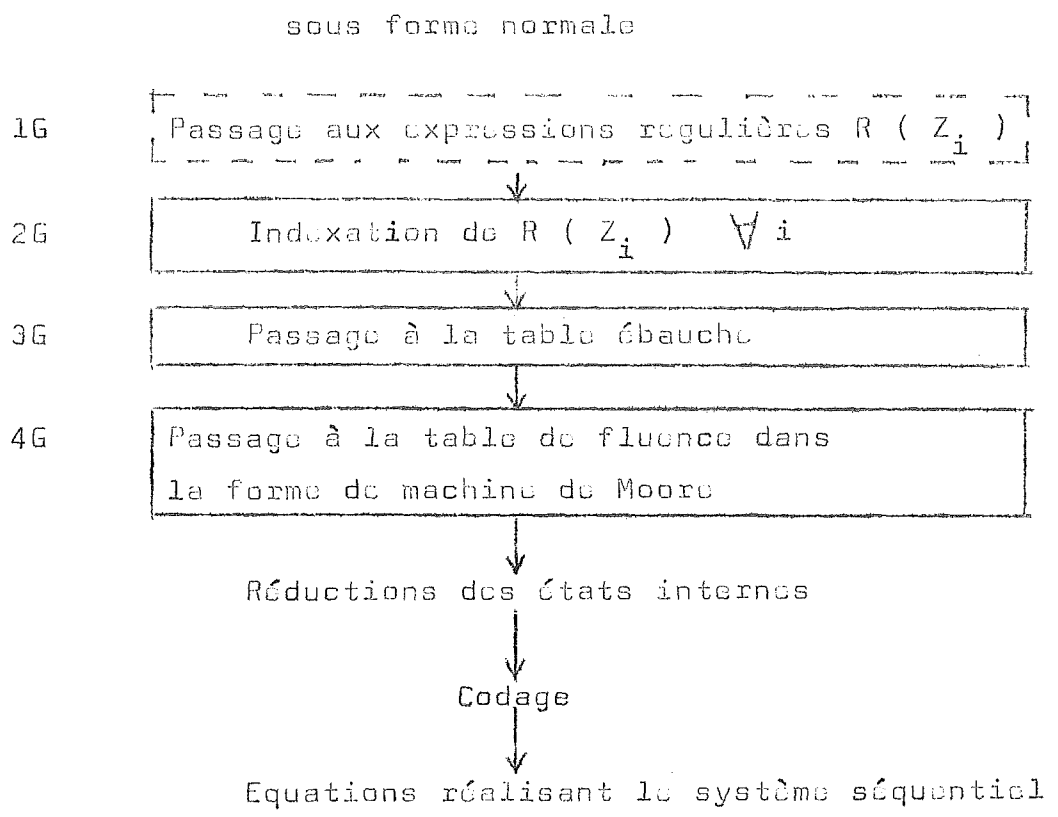


FIG. 1.2

1.41.- Problème envisagé

Nous allons programmer les échelons 2G, 3G et 4G, c'est à dire que l'on supposera déjà réalisée l'étape 1G.

On considèrera un problème donné sous la forme

$$R(Z) = i \quad (P_1 + P_2 + \ldots + P_m)$$

avec $P_m = \nearrow X_j \{X_j\}$ ou $\nearrow$ signifie une séquence de longueur non déterminée et non ordonnée des variables d'entrée.

Il s'agit donc d'un automate asynchrone du type machine transfert qui réalise la reconnaissance de plusieurs séquences d'entrée au moment où elles se présent. C'est le cas, par exemple, de la commande de machines-outils.

Remarque.- Le cas plus général de traitement de plusieurs expressions de la forme R (Z) ci-dessus, autrement dit d'une machine avec un alphabet de sortie supérieur à 1, ne soulève aucune difficulté de principe. Il s'agirait seulement d'introduire dans le programme une simple comparaison entre toutes les expressions du problème.

C H A P I T R E II

LA METHODE DE GLOUSHKOV ET LES SYSTEMES ASYNCHRONES

Le but initial de programmer la méthode de Gloushkov a été changé au cours de nos travaux. Nous avons trouvé un algorithme efficace, plus simple, et puisque il nous paraît valable, nous l'utiliserons à la place de celui-là.

Cet algorithme est dérivé de celui de Gloushkov. C'est à cause de ça que ce chapitre a pour but de tirer sur un exemple quelques remarques utiles qui introduisent le chapitre suivant. Cet exemple va nous permettre aussi mettre en relief les avantages de l 'algorithme qu'on propose.

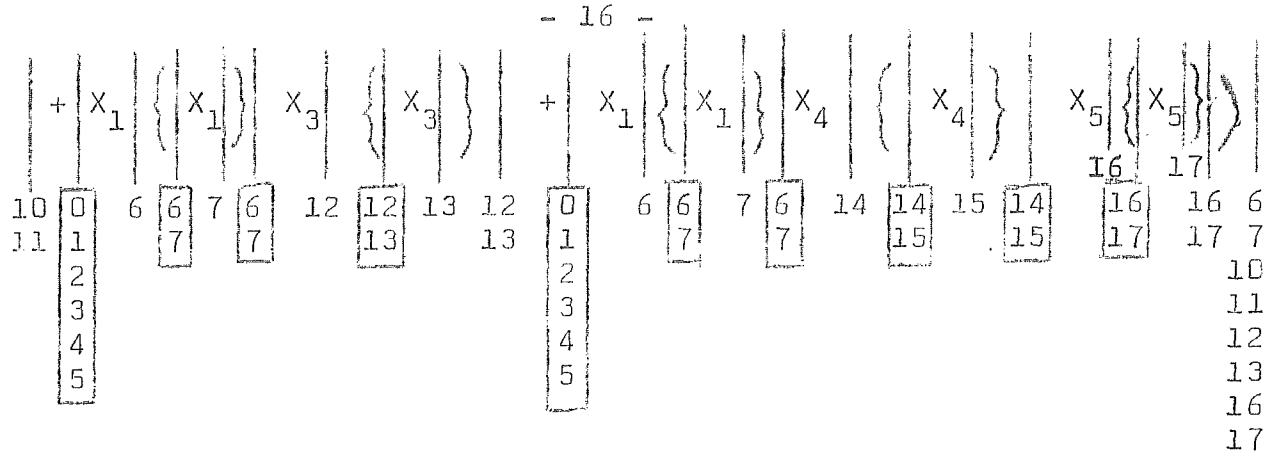
2.1.- Application à un exemple du même type que le problème envisagé

(on peut trouver un exposé de la méthode dans l'ouvrage cité de MM. Perrin, Denouette et Daclin (ref. 2)).

Prenons un exemple simple à 5 variables asynchrones, auquel nous allons appliquer le procédé de Gloushkov jusqu'à l'obtention de la table de fluence.

$$R(Z) = \{X_1\}_0 + \{X_2\}_1 + \{X_3\}_2 + \{X_4\}_3 + \{X_5\}_4 \{X_1 \{X_1\}_6\}_5 + \{X_2 \{X_2\}_8\}_7 X_1 \{X_1\}_{10}$$

0 : 0 1 2 3 4 5 — 1 : 0 1 2 3 4 5 — 2 : 0 1 2 3 4 5 — 3 : 0 1 2 3 4 5 — 4 : 0 1 2 3 4 5 — 5 : 0 1 2 3 4 5 — 6 : 6 7 — 7 : 0 1 2 3 4 5 — 8 : 8 9 — 9 : 8 9 — 10 : 10 11 — 11



On obtient à partir de l'expression indexée la table de la figure 2.1:

	Z	X_1	X_2	X_3	X_4	X_5	
0	-	1,6	2,8	3	4	5	a
1	-	1,6	2,8	3	4	5	b
2	-	1,6	2,8	3	4	5	c
3	-	1,6	2,8	3	4	5	d
4	-	1,6	2,8	3	4	5	e
5	-	1,6	2,8	3	4	5	f
1,6	Z	1,6,7	2,8	3,12	4,14	5	g
2,8	-	1,6,10	2,8,9	3	4	5	h
3,12	Z	1,6	2,8	3,13	4	5	i
4,14	-	1,6	2,8	3	4,15	5,16	j
1,6,7	Z	1,6,7	2,8	3,12	4,14	5	k
2,8,9	-	1,6,10	2,8,9	3	4	5	l
1,6,10	Z	1,6,7,11	2,8	3,12	4,14	5	m
3,13	Z	1,6	2,8	3,13	4	5	n
4,15	-	1,6	2,8	3	4,15	5,16	o
5,16	Z	1,6	2,8	3	4	5,17	p
1,6,7,11	Z	1,6,7,11	2,8	3,12	4,14	5	q
5,17	Z	1,6	2,8	3	4	5,17	r

FIG. 2.1

Première table " ébauche "

Cette première table peut se simplifier. En effet:

- Les 6 premières lignes sont identiques. Posons donc

$$0 \equiv 1 \equiv 2 \equiv 3 \equiv 4 \equiv 5 = \underline{1}$$

- Les lignes g et k sont identiques. Donc 1,6 ≡ 1,6,7; donc 6 ≡ 7 = 2
- Les lignes h et l le sont aussi ⟶ 2,8 ≡ 2,8,9 ⟶ 8 ≡ 9 = 3
- Les lignes m et q sont les mêmes ⟶ 10 ≡ 11 = 4
- De même pour les lignes i et n, j et o, p et r ⟶

$$12 \equiv 13 = \underline{5}$$
$$14 \equiv 15 = \underline{6}$$
$$16 \equiv 17 = \underline{7}$$

La table " ébauche " devient la table de la figure 2.2:

	Z	X_1	X_2	X_3	X_4	X_5
1	-	1,2	1,3	1	1	1
1,2	Z	1,2	1,3	1,5	1,6	1
1,3	-	1,2,4	1,3	1	1	1
1,5	Z	1,2	1,3	1,5	1	1
1,6	-	1,2	1,3	1	1,6	1,7
1,2,4	Z	1,2,4	1,3	1,5	1,6	1
1,7	Z	1,2	1,3	1	1	1,7

FIG. 2.2

qui peut encore se réduire si l'on admet que les lignes 2 et 6 sont égales si les classes d'indices (1,2) et (1,2,4) le sont (figure 2.3):

	Z	X_1	X_2	X_3	X_4	X_5
1	-	1,2	1,3	1	1	1
1,2	Z	1,2	1,3	1,5	1,6	1
1,3	-	1,2	1,3	1	1	1
1,5	Z	1,2	1,3	1,5	1	1
1,6	-	1,2	1,3	1	1,6	1,7
1,7	Z	1,2	1,3	1	1	1,7

Fig.2.3

Evidemment on pourrait aussi identifier les états 1 et 1,3. Nous ne le faisons pas car il s'agit là d'un cas trop particulier et il ne nous interesse pas pour le moment de le détacher (voir paragraphe 3.5).

On obtient finalement la table de la figure 2.4.

	Z	X_1	X_2	X_3	X_4	X_5
A	-	B	C	A	A	A
B	Z	B	C	D	E	A
C	-	B	C	A	A	A
D	Z	B	C	D	A	A
E	-	B	C	A	E	F
F	Z	B	C	A	A	F

Fig. 2.4

Table de fluence

2.11.- Quelques conclusions pratiques

Cet exemple de résolution complète d'un problème permet de tirer un certain nombre de conclusions à propos de la méthode de Gloushkov. Quelques unes sont générales, d'autres attachées au problème envisagé. Les voici:

1) Conséquences générales:

a) L'indexation (même pour un petit problème) est lourde quand on la fait à la main et il faut faire très attention pour ne pas se tromper.

b) Lors de l'application de la méthode à une expression avec des parenthèses d'itération on tombe directement sur des tables redondantes. Ceci traduit la propiété suivante: Les l lignes correspondantes à la réception d'une séquence itérée simple de longueur l manifestent la propriété de superposition indefinie dans le temps.

En particulier, pour un automate asynchrone traité de cette façon-ci, la première table " ébauche " n'est pas du tout physique, en ce sens qu'il ne présente pas un état stable pour chaque état de départ. Tout est mêlé, si l'on peut dire.

c) La première table " ébauche " a un nombre de lignes égal à l'indice de plus grand poids plus un.

2) Conséquences dûes au problème envisagé:

d) Chaque case de la première table " ébauche " peut avoir un nombre d'indices égal au nombre des vecteurs de la séquence la plus longue plus un.

e) Tous les indices qui apparaissent dans l'événement universel appartiennent au même ensemble. Si l'on note cet ensemble par un seul chiffre (un 1 par exemple) on voit bien que, à ce moment-là, l'événement universel sera représenté dans la table " ébauche " par un 1 dans toutes les cases de la première ligne. Et ceci paraît logique puisque après tout la fonction du dit événement est de placer le début de la reconnaissance à un instant t_1 indéterminé.

f) Les indices qui apparaissent pour la première fois dans la parenthèse normale peuvent se simplifier deux à deux.

2.2.- Programmation du problème

Tel que le problème est posé il faut donc prevoir les phases suivantes pour la programmation:

- indexation de l'expression regulière
  - repérage et indexation des endroits fondamentaux
  - repérage et indexation des endroits non fondamentaux
- passage à la table de fluence
  - classement des indices en fondamentaux et antefondamentaux
  - obtention de la table " ébauche "
  - simplification de la table " ébauche " et passage à la table de fluence.

A noter dans ces grandes lignes que le programme d'indexation doit être théoriquement beaucoup plus difficile que le programme de la table. Il doit suivre fidèlement les six reglès d'extension d'indices de Gloushkov. Pour le programme de la table il ne s'agit finalement que d'un classement de nombres.

En particulier il faudrait associer à chaque endroit N + 1 memoires de la machine (N est le nombre des vecteurs de l'alphabet d'entrée), donc variable pour chaque problème, pour ranger les indices des endroits antéfondamentaux de l'événement universel et homogénéiser le traitement. Ceci amène à un gaspillage considérable des memoires de la machine et à une perte de temps puisqu'il faut les explorer toutes à chaque coup.

Il faudrait aussi remplacer les accolades par un autre symbole que la machine puisse reconnaître.

Nous avons écrit un organigramme très detaillé seulement pour l'indexation qui ne comporterait dans les meilleurs des cas pas moins de 450 - 500 instructions en langage machine pour CAB - 500. Finalement nous l'avons abandonné.

En effet, les conséquences e) et f) du paragraphe 2.11 nous ont poussé à abandonner le chemin classique de Gloushkov au profit de quelques modifications que nous allons introduire dans le chapitre 3. Ces modifications auront à notre avis des avantages absolument remarquables pour la synthèse des automates asynchrones par la méthode des expressions régulières, pour l'algèbre des itérations dans certains cas et en particulier pour la programmation de notre problème.

C H A P I T R E III

QUELQUES CONTRIBUTIONS A LA THEORIE DES AUTOMATES

Ce chapitre et les suivants présentent deux aspects simultanés: d'une part la proposition et réalisation sur machine d'une nouvelle méthode algorithmique et l'exposé de quelques autres propriétés. D'autre part sa comparaison avec la méthode de Gloushkov.

3.1.- Une nouvelle notation des expressions itérées

Si l'on admet (réf. 4) que:

$$P^0 = \text{séquence de longueur nulle}$$
$$P^1 = P$$
$$P^2 = P.P$$
$$P^3 = P.P.P$$
$$\ldots\ldots\ldots\ldots$$
$$P^n = \underbrace{P.P\ldots.P}_{n \text{ fois}}$$

et que $\left\{ P \right\} = P^*$

nous proposons d'admettre aussi ce qui suit:

$$P \left\{ P \right\} = P^{1*}$$
$$\left\{ P \right\} P = P^{*1}$$

Ceci permet d'écrire autrement les propriétés des expressions itérées du paragraphe 1.31:

$$P^{**} = P^*$$
$$P^* = P^0 + P^{1*}$$
$$P^{1*} = P^{*1}$$
$$P^* P^* = P^*$$
$$P^* + P = P^*$$

3.11.- Cas particulier: les variables asynchrones

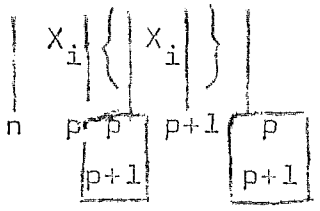
Elles s'exprimeront comme ci-dessous

$$X\left\{ X \right\} \longrightarrow X^{1*}$$

De notre point de vue l'élimination des accolades serait arbitraire si elle n'était pas valable au moment d'appliquer le procédé d'indexation de Gloushkov ou un algorithme qui amène aux mêmes resultats.

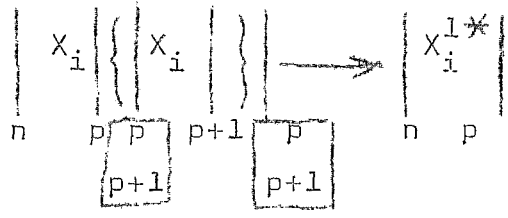
3.2.- Proposition d'un nouvel algorithme derivé de celui de Gloushkov pour les machines asynchrones

En appliquant les règles d'indexation à une commande asynchrones $X_i \left\{ X_i \right\}$ on arrive à

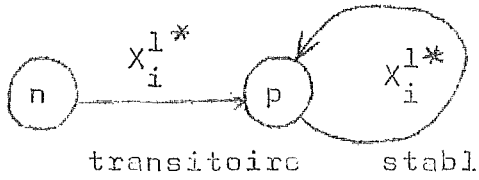


où p et p + 1 appartiennent au même ensemble d'indices puisqu'ils amènent par application des mêmes vecteurs d'entrée au même état de la machine (voir conséquence f) du paragraphe 2.11)

Nous avons donc le droit d'écrire



avec la signification explicitée sur le graphe de la figure 3.1.



3.21.- Propriétés

Cet algorithme est beaucoup plus souple que celui de Gloushkov parce que:

- Il ne fait pas de distinction entre des endroits fondamentaux et antéfondamentaux. Ou si l'on préfère chaque endroit devient une fusion de deux.

- Il place le traitement des problèmes asynchrones au même rang de simplicité que celui des problèmes synchrones. En particulier l'indexation se fait exactement de la même façon dans les deux cas.

- Au passage à la table, dans la formation de laquelle il faut tenir compte de la signification énoncée plus haut, on tombe directement sur une table non redondante très réduite. Cette table a, en plus, un sens physique car on y trouve un état stable par ligne.

- Sa souplesse permet facilement d'exploiter plus à fond dans l'indexation quelques propriétés de l'algèbre des expressions regulières (voir 3.4) qui réduisent davantage la table.

- Dans le cas le plus général il est toujours susceptible d'être amené à la manipulation d'un seul indice par endroit.

Une conséquence générale de tout ce qui vient d'être dit est l'élargissement considérable des limites des problèmes susceptibles d'être traités à la main aussi bien que par calculateur.

3.22.- Généralisation à une expression de la forme P^{1*}

Dans une expression de la forme $P\{P\}$ chaque indice de l'expression itérée appartient au même ensemble que ceux qui sont placés à l'endroit homologue de l'expression non itérée. Ceci découle de la propriété $P\{P\} = \{P\}P$.

Nous en tirons comme conséquence qu'on pourra écrire

$$P\left|\left\{\ P\ \right\}\right|_{\alpha'}\rightarrow P^{1*}\Big|_{\alpha}$$

où les indices de l'ensemble final α font partie de l'ensemble des indices de tous les endroits initiaux de l'expression régulière P. Par conséquent, l'indexation deviendra:

- Application des règles d'indexation de Gloushkov ou application de l'algorithme du paragraphe 3.2 si c'est le cas à l'interieur de l'expression P.
- Ensuite étendre chaque indice final à l'endroit initial correspondant.

Nous rejetons en annexe un exemple d'application de ce que nous venons de dire.

Remarque.-

Cette généralisation est valable pourvu que tous les indices de l'expression P n'appartiennent pas à d'autres ensembles que ceux y définis.

3.3.- Autres propriétés des expressions itérées

Dans une expression $\left|\left\{\ P\ \right\}\right|_{\alpha'}^{\alpha} = P^{*}$ on peut trouver un ou plusieurs indices de α' qui appartiennent à l'ensemble initial α. Aux indices qui remplissent cette condition on peut associer un même numéro, ce qui simplifie au départ la table.

En particulier, dans certains cas tous les indices des endroits finals peuvent faire partie de l'ensemble initial α.

On aurait donc dans ce cas

$$\left|\left\{\ P\ \right\}\right|_{\alpha'}^{\alpha}\rightarrow\left|P^{*}\right|_{\alpha}^{\alpha}$$

Exemple.- L'expression

$$R(Z) = \left|\left\{\left|X_1\right|\left|X_2\right|+\left|X_1\right|\left|X_2\right|\left|X_3\right|+\left|X_3\right|\left|X_2\right|\left|X_3\right|\right\}\right|$$

$$\begin{array}{cccccccccccccc} 0 & 0 & 1 & 2 & 0 & 1 & 2 & 3 & 0 & 4 & 5 & 6 & 0 \\ & 2 & & & 2 & & & & 2 & & & & 2 \\ & 3 & & & 3 & & & & 3 & & & & 3 \\ & 6 & & & 6 & & & & 6 & & & & 6 \end{array}$$

devient, en faisant (0,3,6) $\longrightarrow$ 0, 1 $\longrightarrow$ 1, 2 $\longrightarrow$ 2, 4 $\longrightarrow$ 3 et 5 $\longrightarrow$ 4 :

$$R(Z) = \left|\left\{\left|X_1\right|\left|X_2\right| + \left|X_1\right|\left|X_2\right|\left|X_3\right| + \left|X_3\right|\left|X_2\right|\left|X_3\right|\right\}\right|$$
$$\begin{matrix} 0\ 0 & 1 & 2\ 0 & 1 & 2 & 0\ 0 & 3 & 4 & 0\ 0 \\ 2 & & 2 & & & 2 & & & 2 \end{matrix}$$

Dans cet exemple on réduit au départ de deux lignes une table qui en a sept.

A noter que ceci peut se faire systematiquement.

3.31.- $\underline{\left\{X\right\} = X^*}$

Cas particulier du précédent: dans les mêmes cas on pourrait écrire

$$\left|\left\{X\right\}\right|_{\alpha\ \alpha'} = \left|X^*\right|_{\alpha\ \alpha}$$

Cette propriété peut être interessant dans certains cas. (Voir annexe)

3.32.- <u>Evénement universel $\left\{i\right\} = i^*$</u>

C'est aussi un cas particulier, mais là les indices de tous les endroits appartiennent au même ensemble. Il n'y a pas de condition limitative.

Cette propriété tout à fait générale, qui avait été dejà empiriquement amorcée dans le point e) du paragraphe 2.11, place l'événement universel sous un nouveau jour que nous résumons comme suit:

- Il améne à une table non initiale (instant indéterminé)
- Sa présence se notera par un 1 dans toutes les cases de la première ligne (correspondant à l'état 1) de la table " ébauche ".
- Pour garder présent dans l'esprit qu'on manipule un problème avec événement universel il peut être utile de le conserver dans l'écriture sous la forme suivante: $\left|i^*\right|_{1\ 1}$.

- Mais si ce n'est pas necessaire on peut même s'en passer tranquillement et ne pas l'écrire (c'est le cas de notre problème).

3.4.- Indexation dans le sens inverse

Avant de passer à la résolution d'un exemple nous allons profiter d'une propriété de l'algèbre des expressions regulières pour introduire l'indexation en sens inverse. En effet, lorsqu'on a une expression régulière sous la forme ($P_1 + P_2 + \ldots + P_j + \ldots + P_m$) où $P_j = X_j$ (cas synchrone) ou bien $P_j = X_j^{1*}$ (cas asynchrone) et qu'il existe des paires de paquets qui peuvent se mettre sous la forme RS + MS = (R + M)S on aboutit à la conséquence pratique suivante:

En comparant entre eux tous les paquets lettre par lettre et en sens inverse (de la droite vers la gauche) on peut associer les mêmes indices aux endroits qui suivent des lettres identiques. Ceci en ce qui concerne les machines asynchrones. Pour les machines synchrones c'est la même chose mais appliquée aux endroits qui suivent et qui précèdent les lettres identiques. comprobado

Puisque, dans une grande mesure, l'indexation normale ou dans le sens direct est basée sur la propriété SR + SM = S(R + M) d'où découle finalement un regroupement partiel et une diminution du nombre d'indices utilisés (donc une réduction au départ de la machine) il nous paraît assez raisonnable d'indexer aussi dans le sens contraire pour réduire davantage ce nombre.

Comme illustration à ce que nous venons de dire considérons une expression régulière quelconque du type qui nous intéresse:

$$R(Z) = \left| i^* \right| \left(\left| X_1^{1*} \right| X_2^{1*} \left| X_3^{1*} \right| X_4^{1*} \left| X_5^{1*} \right| + \left| X_2^{1*} \right| X_1^{1*} \left| X_3^{1*} \right| X_4^{1*} \left| X_5^{1*} \right|$$

$$1 \quad 1 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 1 \quad 7 \quad 8 \quad \not{9}\ 4 \quad \not{10}\ 5 \quad \not{11}\ 6$$

$$+\left|\begin{matrix} X_3^{1*} & X_1^{1*} & X_2^{1*} & X_4^{1*} & X_5^{1*} \\ 1 & 12 & 13 & 14 & \cancel{15}\ 5 \end{matrix}\right| \left|\begin{matrix}) \\ \cancel{16}\ 6 \end{matrix}\right| \quad 6$$

Ce problème, qui traité par la méthode de Gloushkov aurait abouti à une première table " ébauche " de 36 lignes, par notre méthode aboutit à une de 17 lignes et en indexant en plus dans le sens inverse à une de 11 lignes.

3.5.- Application: exemple du même type que le problème envisagé

On reprend l'exemple du paragraphe 2.1 que l'on va traiter par la méthode qu'on vient de proposer. Nous nous passons de l'écriture de i^*. Alors l'expression s'écrit

$$\left|\begin{matrix} X_1^{1*} \\ 1\quad 2 \end{matrix}\right| + \left|\begin{matrix} X_2^{1*} & X_1^{1*} \\ 1 & 3 \end{matrix}\right| + \left|\begin{matrix} X_1^{1*} & X_3^{1*} \\ \cancel{4}\ 2 & 5 \end{matrix}\right| + \left|\begin{matrix} X_1^{1*} & X_4^{1*} & X_5^{1*} \\ 1 & 2 & 6 \quad 7 \end{matrix}\right|$$

On obtient la table 3.2

	Z	X_1	X_2	X_3	X_4	X_5
1	-	1,2	1,3	1	1	1
1,2	Z	(1,2)	1,3	1,5	1,6	1
1,3	-	1,2	(1,3)	1	1	1
1,5	Z	1,2	1,3	(1,5)	1	1
1,6	-	1,2	1,3	1	(1,6)	1,7
1,7	Z	1,2	1,3	1	1	(1,7)

Fig. 3.2

La comparaison est concluante. On peut faire l'indexation sans erreur et on aboutit d'un seul coup à une table

minimale dans la plupart des cas, presque minimale dans quelques uns. D'ailleurs, dans la pratique du travail, on peut éviter l'utilisation des exposants 1^*, si l'on veut.

Remarque.-

On aurait pu dans ce cas appliquer la propriété du paragraphe 3.31. Voyons comment:

Les deux premiers paquets deviennent

$$\left|\begin{matrix} X_1^{1*} \\ 1 \end{matrix}\right| + \left|\begin{matrix} X_2^{1*} \\ 2 \end{matrix}\right| \left|\begin{matrix} X_1^{1*} \\ 1 \end{matrix}\right| = \left(\left|\begin{matrix} X_2^{o} \\ 2 \end{matrix}\right| + \left|\begin{matrix} X_2^{1*} \\ 1 \end{matrix}\right|\right) \left|\begin{matrix} X_1^{1*} \\ 1 \end{matrix}\right| = \left|\begin{matrix} X_2^{*} \\ 2 \end{matrix}\right| \left|\begin{matrix} X_1^{1*} \\ 1 \end{matrix}\right|$$

car l'indice 3 n'apparaît nulle part ailleurs dans l'expression. Ceci explique la possibilité de faire encore (1,3) $\longrightarrow$1. En annexe on trouvera un cas très curieux mais vraiment trop peu probable.

C H A P I T R E IV

PROGRAMMATION

4.1.- Données du problème

- Expression régulière sous la forme $P_1 + P_2 + \ldots + P_j$. Les variables asynchrones seront notées $X_1^{1*} = A$; $X_2^{1*} = B$; etc...
- N, nombre des vecteurs de l'alphabet d'entrée.
- c_{max}, nombre des paquets de séquences.
- n_{max}, nombre des symboles de l'expression.

4.2.- Occupation des mémoires du calculateur CAB-500

Bloc A (48,0) $A_1 = A+1$, $A_2 \ldots A_{n_{max}}$ ⟶ expression régulière

" B (50,0) $B_1 = B+1$, $B_2 \ldots B_{n_{max}+1}$ ⟶ indices

" C (52,0) $C_1 = C+1$, $C_2 \ldots C_{c_{max}}$ ⟶ indices finals

" D (53,0) $D_1 = D+1$, $D_2 \ldots D_{d(N+2)+g} \ldots$ ⟶ Table

" E (52,20) $E_1 = E+1$, $E_2 \ldots E_e \ldots$ ⟶ Indices codifies de chaque case de la table

" F (52,50) $F_1 = F+1$, $F_2 \ldots F_f \ldots$ ⟶ Indices decodifies de chaque E_c

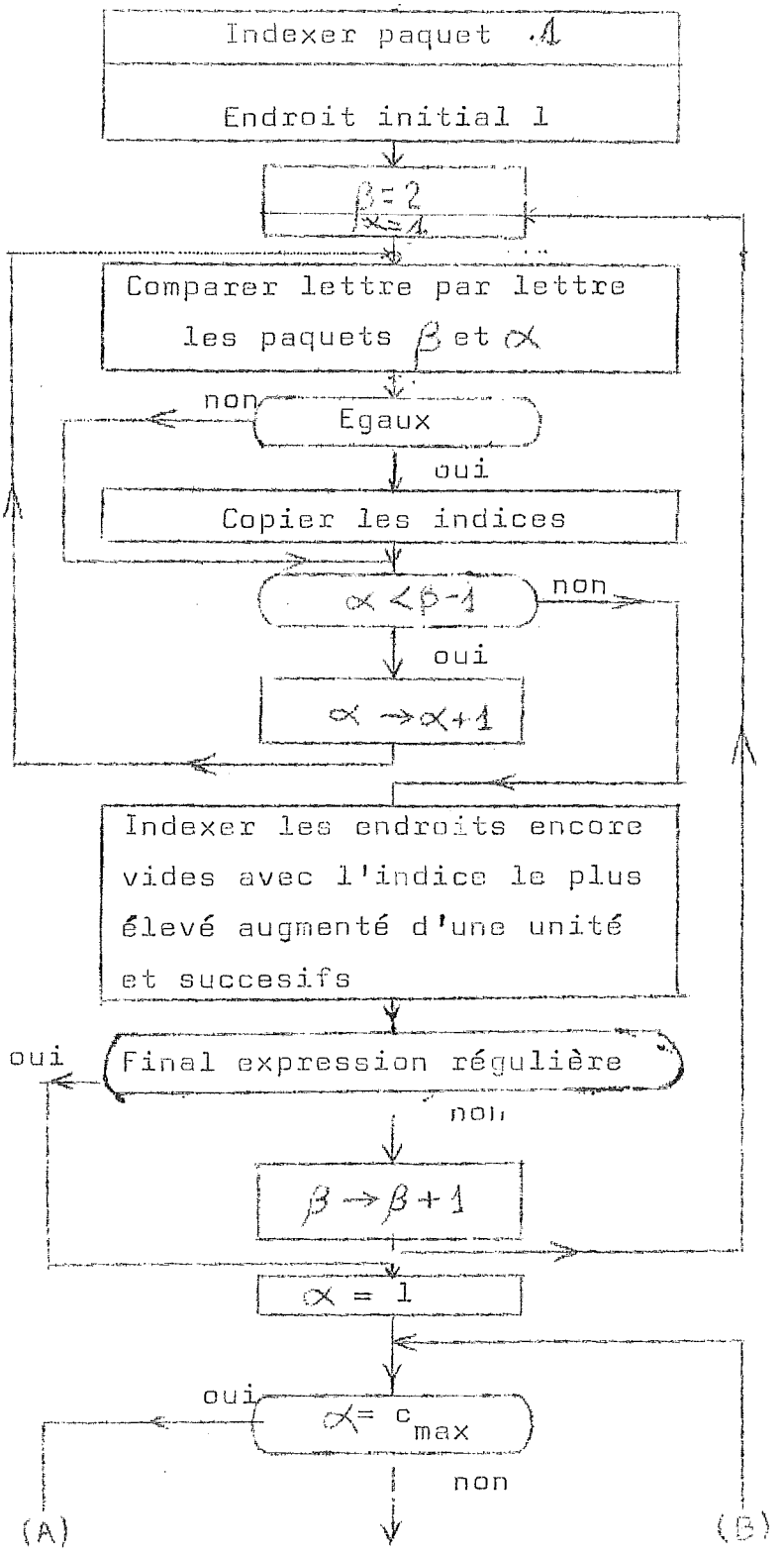
" L (52,80) $L_1 = L+1$, $L_2 \ldots L_l \ldots L_N$ ⟶ Vecteurs d'entrée

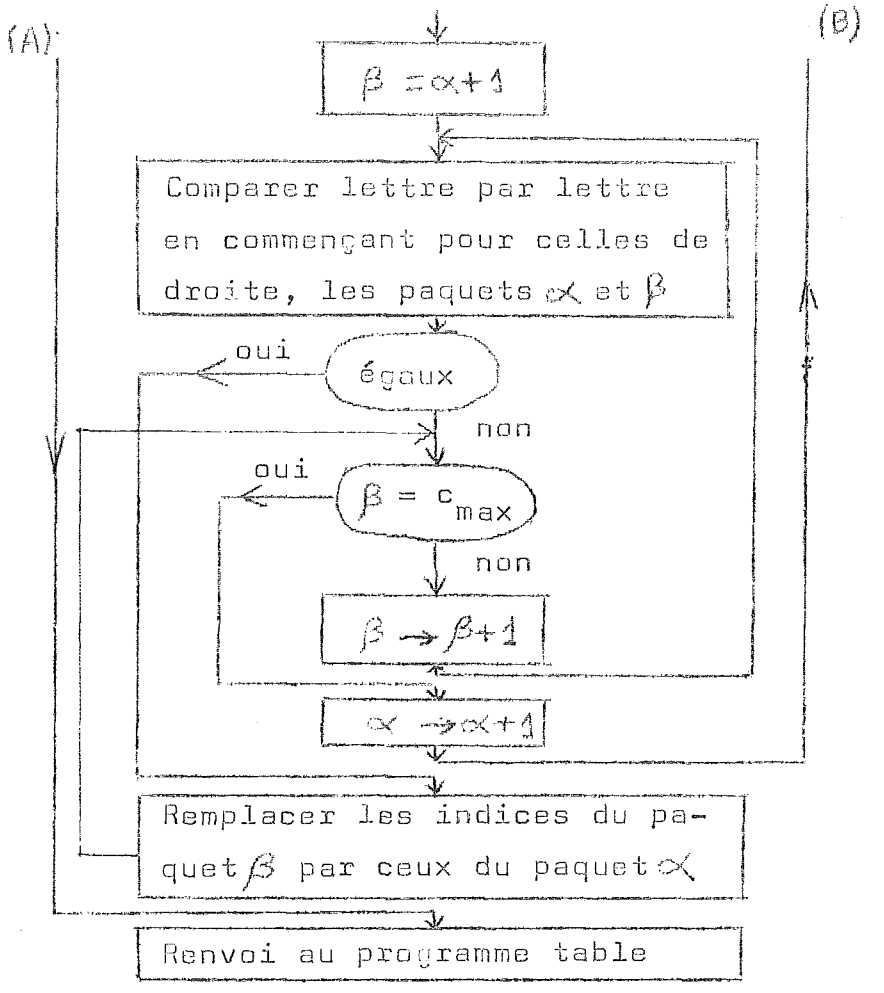
" Q (52,110) $Q_o = Q$, $Q_1 = Q+1$, $Q_2 \ldots Q_9$ ⟶ Diverses

Mémoires 43,51; 43,52; ... 43,82 ⟶ Puissances de 2

Au total on a besoin de $2n_{max} + 2c_{max} + 33N + 75$ mémoires de la machine.

4.3.- Programme d'indexation. Macro-organigramme





4.31.- Symboles utilisés dans le micro-organigramme

La programmation a été faite en langage machine (réf. 5) (voir annexe).

nº de registre	symbole	signification
R_1	c	comptage des paquets β
R_2	n	progression symbole par symbole dans l'indexation
R_3	f	dernier indice utilisé
R_4	+	
R_5	n_{max}	
R_6	d	comptage des paquets α
R_7	k	progression symbole par symbo-

le dans la comparaison

R_0 i comptage des symboles explorés

Les parties de l'organigramme qui sont en pointillé sont independentes du programme d'indexation. On les a incorporées dedans mais ilserait aussi bien les placer dans le programme de la table.

Le programme d'indexation est valable aussi pour les problèmes synchrones. Il n'a aucune limite de fonctionnement si ce n'est la capacité de mémoires de la machine.

4.4.- Programme de la table. Macro-organigramme

Pour aboutir à une utilisation compacte des mémoires de la machine nous nous sommes servis d'une astuce que nous a été communiquée par M. J. P. Perrin du C.E.R.A et qui consiste à ranger dans une seule mémoire tous les indices d'une même classe ou ce qui revient au même tous les indices d'une case de la table. On établit la correspondence suivante:

$$0 \longrightarrow 2^0$$
$$1 \longrightarrow 2^1$$
$$2 \longrightarrow 2^2$$
$$\ldots\ldots$$
$$n \longrightarrow 2^n$$
$$\ldots\ldots$$

de façon qu'un indice n sera representé par un bit dans la position n des 32 qui possede une mémoire. Ainsi par exemple la classe 1,2,6 se placera dans une mémoire codée comme voici

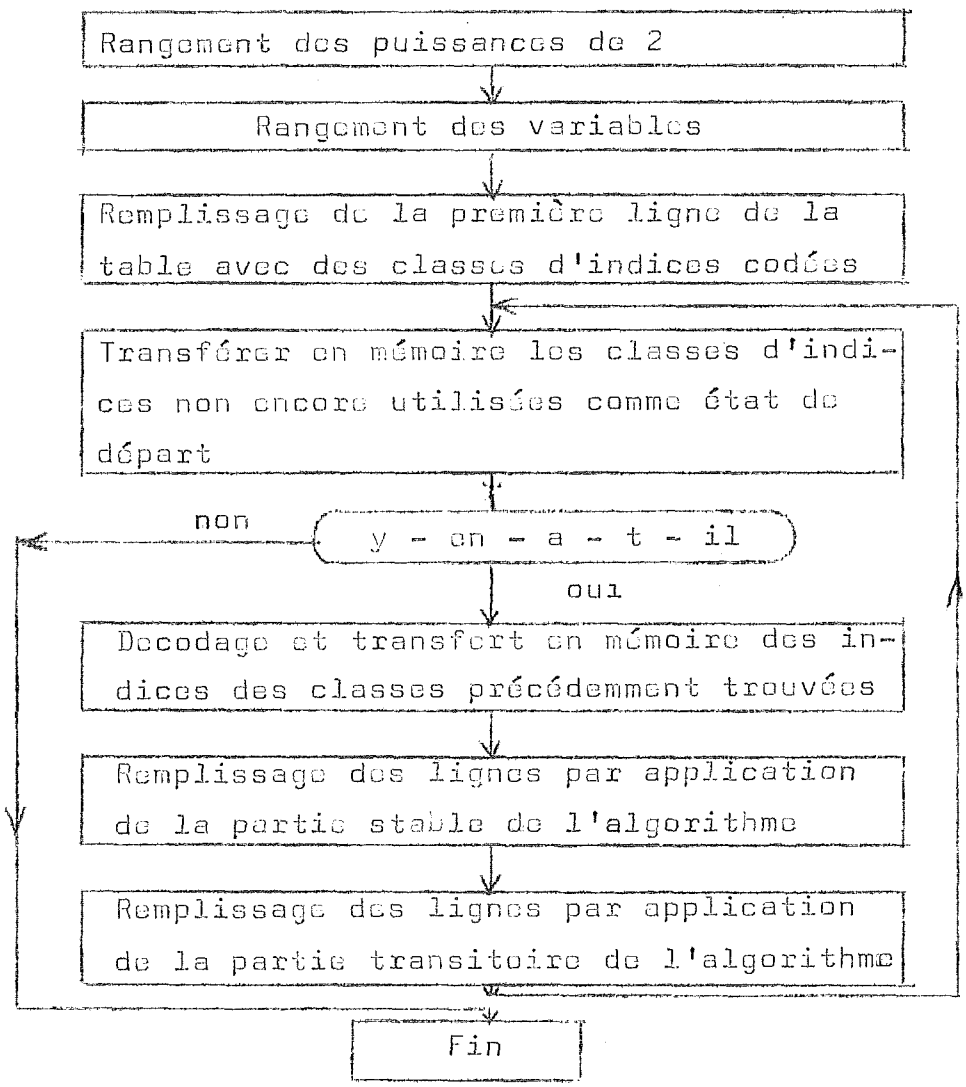
----etc------1---11-

et elle sortira un numéro $2^1 + 2^2 + 2^6 = 70$. Donc on obtiendra directement la table de fluence.

Pour la colonne de la sortie la machine sortira 0 ou 1 selon l'événement non réalisé ou réalisé.

Les parties du programme correspondant au rangement des puissances de 2 et des variables ont été incluses dans le programme d'indexation au lieu de le faire dans le programme de la table.

Il est évident qu'on introduit une limitation du point de vue du volume du problème à traiter. Le programme de la table se limite à traiter les problèmes jusqu'à un indice maximum égal à 31. Nous avons prévu dans le programme d'indexation un test de securité.



Le programme se met en fin. Il faut demander à la machine d'écrire la table.

4.41.- Symboles utilisés dans le micro-organigramme

(Voir annexe)

nº registre	Symbole	Signification
R_1	c	divers comptages
R_2	n	progression symbole par symbole et indice par indice
R_3	f, p	comptage pour l'exploration et rangement des indices decodés
R_4	g	reperage des cases de chaque ligne
R_5	c, x	comptage pour le rangement et l'exploration des classes d'indices codées.
R_6	d, r	comptage et repérage des lignes de la table
R_7	m	comptage pour l'exploration des états de départ
R_8	l	comptage pour l'exploration des variables.

4.42.- Utilisation des mémoires Q

Q_o ------ c_{max}
Q_1 ------ n_{max}
Q_2 ------ N
Q_3 ------ d (programme table)
Q_4 ------ N + 2
Q_5 ------ c
Q_6 ------ f (progr. table)
Q_7, Q_8 ------ k
Q_9 ------ i

C H A P I T R E V

CONCLUSION

5.1.- Sur le plan théorique nous avons proposé un nouvel algorithme pour le traitement des automates asynchrones qui, pensons nous, grâce à sa simplicité doit pouvoir remplacer la méthode de Gloushkov. En particulier l'indexation devient tout à fait semblable à celle des problèmes synchrones.

Nous ameliorons aussi cette indexation en introduisant ce que nous appelons l'indexation dans le sens inverse qui, comme nous venons de dire, est à la fois applicable aux problèmes du type asynchrone et synchrone.

Tout ceci fait aboutir d'un seul coup à une table minimale ou quasiminimale. On peut se passer, donc, des ennuyeuses operations de réduction du nombre des lignes de la table.

Tous ces résultats débordent, pensons-nous, largement les limites que nous nous étions fixés dans le titre de cette thèse. Ils sont géneraux.

5.2.- Nous avons trouvé en plus d'autres résultats théoriques dans le cadre des expressions itérées qui peuvent dans certains cas être fort utiles, ainsi que nous le montrons en annexe.

En particulier nous pensons avoir ramené l'événement universel à la place qui lui convient, beaucoup moins importante qu'elle ne le paraissait.

Les résultats que nous résumons dans ce paragraphe sont communs aux systèmes synchrones et asynchrones.

5.3.- Pour laprogrammation qui était finalement le but de notre thèse, nous avons suivi la méthode proposée. Les programmes sont énormement plus simples que ceux qui résultent de l'application de la méthode de Gloushkov. Les accolades et l'événement universel ont disparu, le nombre des autres symboles s'est réduit de moitié et chose importante, il n'existe qu'un seul indice par endroit.

Le programme d'indexation est aussi valable pour un système synchrone.

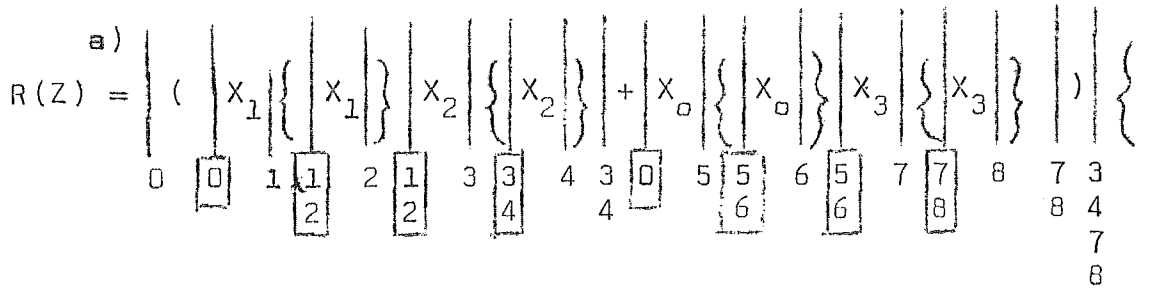
Dans le programme de la table il n'a pas été nécessaire d'introduire une réduction du nombre de lignes puisqu'il amène directement à une table de fluence minimale ou presque minimale.

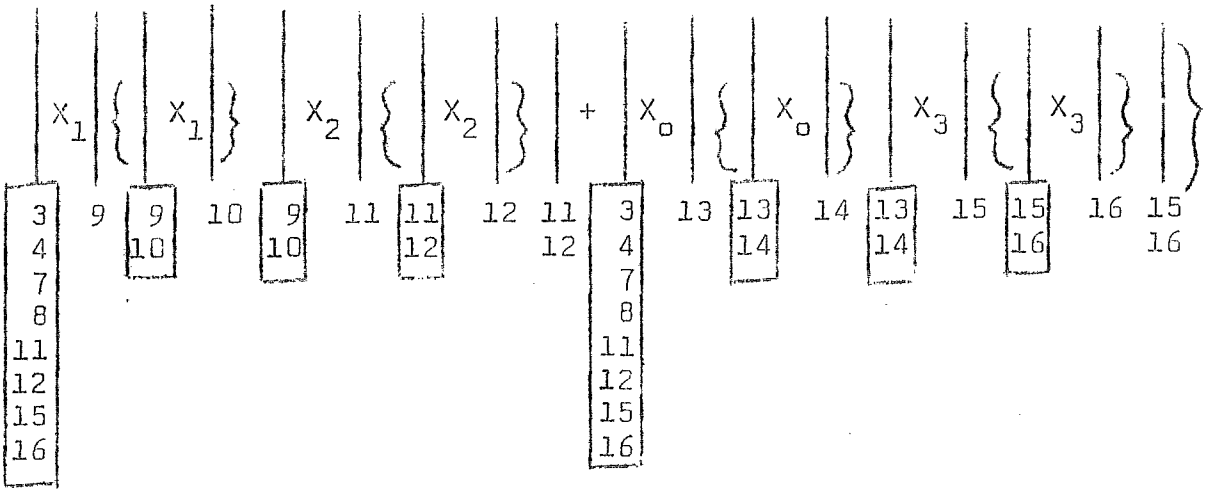
A N N E X E I

EXERCICES COMPARES

Nous allons donner les exercices résous d'abord par la méthode de Gloushkov, ensuite au moyen de l'algorithme et des simplifications du chapitre III.

a)





Cet expression, de la forme P { P }, a comme endroits finals (11, 12, 15, 16, 3, 4, 7 et 8).

		Z	X_0	X_1	X_2	X_3
	0	-	5	1	-	-
1	1	-	-	2	3	-
	2	-	-	2	3	-
2	3	Z	13	9	4	-
	4	Z	13	9	4	-
3	5	-	6	-	-	7
	6	-	6	-	-	7
4	7	Z	13	9	-	8
	8	Z	13	9	-	8
5	9	-	-	10	11	-
	10	-	-	10	11	-
6	11	Z	13	9	12	-
	12	Z	13	9	12	-
7	13	-	14	-	-	15
	14	-	14	-	-	15
8	15	Z	13	9	-	16
	16	Z	13	9	-	16

première table
" ébauche "

	Z	X_0	X_1	X_2	X_3
0	-	3	1	-	-
1	-	-	(1)	2	-
2	Z	7	5	(2)	-
3	-	(3)	-	-	4
4	Z	7	5	-	(4)
5	-	-	(5)	6	-
6	Z	7	5	(6)	-
7	-	(7)	-	-	8
8	Z	7	5	-	(8)

deuxième table

La deuxième table peut se réduire avec les demarches suivantes:

- la ligne du 2 est égale à celle du 6 si 2 = 6
- la ligne du 4 est égale à celle du 8 si 4 = 8
- la ligne du 3 est égale à celle du 7 si 3 = 7 et 4 = 8
- la ligne du 1 égale à celle du 5 si 1 = 5 et 2 = 6

On aboutit donc à la table réduite

	Z	X_o	X_1	X_2	X_3
0	-	3	1	-	-
1	-	-	(1)	2	-
2	Z	3	1	(2)	-
3	-	(3)	-	-	4
4	Z	3	1	-	(4)

---

a') L'expression du problème devient la suivante

$$R(Z) = P^{1*} = \left| \left(\left| X_1^{1*} \right|_{0\,[0\,2\,4]} \left| X_2^{1*} \right|_{[1]\ 2} + \left| X_o^{1*} \right|_{[0\,4\,2]} \left| X_3^{1*} \right|_{[3]\ 4} \right)^{1*} \right|_{2\,4}$$

qui amène directement à la table

	Z	X_o	X_1	X_2	X_3
0	-	3	1	-	-
1	-	-	(1)	2	-
2	Z	3	1	(2)	-
3	-	(3)	-	-	4
4	Z	3	1	-	(4)

b)

Tiré de réf. 2, pp 95 - 97 du même volume.

$$R(Z) = \left\{ X_o \right\} X_1 \left\{ X_1 \right\} X_o \left\{ X_o \right\} X_1 \left\{ X_1 \right\} \left\{ X_o \left\{ X_o \right\} X_1 \left\{ X_1 \right\} X_o \left\{ X_o \right\} X_1 \left\{ X_1 \right\} \right\}$$

Les auteurs donnent la première table que l'on peut obtenir en appliquant le procédé de Gloushkov.

	Z	X_o	X_1
A	-	B	C
B	-	B	C
C	-	E	D
D	-	E	D
E	-	E	D
F	-	F	G
G	Z	J	H
H	Z	J	H
J	-	K	L
K	-	K	L
L	-	P	M
M	-	P	M
P	-	R	S
R	-	R	S
S	Z	K	T
T	Z	K	T

simplifiable à celle-ci

	Z	X_o	X_1
1	-	(1)	2
2	-	3	(2)
3	-	(3)	4
4	Z	5	(4)
5	-	(5)	6
6	-	7	(6)
7	-	(7)	8
8	Z	5	(8)

qui n'est simplifiable que si $4 \equiv 8$

Ils arrivent à la table beaucoup plus réduite:

	Z	X_o	X_1
1	-	(1)	2
2	-	3	(2)
3	-	(3)	4
4	Z	1	(4)

b')

$$R(Z) = \left(\left| X_o^{*} \right| \left| X_1^{1*} \right| \left| X_o^{1*} \right| \left| X_1^{1*} \right| \right)^{1*}$$

1 1 2 3 4 4

4

	Z	X_o	X_1
1	-	(1)	2
2	-	3	(2)
3	-	(3)	4
4	Z	1	(4)

---

c)

$$R(Z) = \left\{ \left| X_1 \right| + \left| X_2 \right| + \left| X_3 \right| \right\} \left(\left| X_1 \right| \left\{ \left| X_1 \right| \right\} + \left| X_2 \right| \left\{ \left| X_2 \right| \right\} \left| X_1 \right| \left\{ \left| X_1 \right| \right\} + \left| X_1 \right| \left\{ \left| X_1 \right| \right\} \left| X_3 \right| \left\{ \left| X_3 \right| \right\} \right| \right)$$

0 1 2 3 0 4 5 4 6 7 7 8 9 8

0 0 0 0 4 5 0 6 7 6 8 9

1 1 1 1 5 1 2 7

2 2 2 2 2 3

3 3 3 3

3

$$+ \left| X_1 \right| \left\{ \left| X_1 \right| \right\} \left| X_3 \right| \left\{ \left| X_3 \right| \right\} \Big| \Big| \Big|$$

4 5 10 11 10 4

0 4 4 4 10 10 5

1 5 5 11 11 8

2 9

3 10

11

	Z	X_1	X_2	X_3
0	-	1,4	2,6	3
1	-	1,4	2,6	3
2	-	1,4	2,6	3
3	-	1,4	2,6	3
1,4	Z	1,4,5	2,6	3,10
2,6	-	1,4,8	2,6,7	3
1,4,5	Z	1,4,5	2,6	3,10
3,10	Z	1,4	2,6	3,11
2,6,7	-	1,4,8	2,6,7	3
1,4,8	Z	1,4,5,9	2,6	3,10
3,11	Z	1,4	2,6	3,11
1,4,5,9	Z	1,4,5,9	2,6	3,11

- On remarque que 0 = 1 = 2 = 3
- (1,4) = (1,4,5)
- (2,6) = (2,6,7) $\longrightarrow$ 6 = 7
- (3,10) = (3,11) puisque 4 = 5 $\longrightarrow$ 10 = 11
- (1,4,8) = (1,4,5,9) puisque 10 = 11 $\longrightarrow$ 8 = 9

Appelons 0 = 1 = 2 = 3 $\longrightarrow$ 1

4 = 5 $\longrightarrow$ 2

6 = 7 $\longrightarrow$ 3

8 = 9 $\longrightarrow$ 4

10 = 11 $\longrightarrow$ 5

$\Downarrow$

	Z	X_1	X_2	X_3
1	-	1,2	1,3	1
1,2	Z	1,2	1,3	1,5
1,3	-	1,2,4	1,3	1
1,5	Z	1,2	1,3	1,5
1,2,4	Z	1,2,4	1,3	1,5

On remarque que (1,2) = (1,5) et que ligne (1,2) = ligne (1,2,4) si (1,2) = (1,2,4).
Donc (1,2) = (1,5) = (1,2,4) = B. A ce moment-là 1 serait identique à (1,3) $\longrightarrow$ 1 = 1,3 = A

$\Downarrow$

	Z	X_1	X_2	X_3
A	-	B	A	A
B	Z	B	A	B

c')

Celui-ci est vraiment un cas trop particulier où l'on peut appliquer en plus de l'algorithme pour les variables asynchrones, de la propriété de l'événement universel et de l'indexation dans le sens inverse, la propriété du paragraphe 3.31, en tenant compte du fait que

$$X_i^0 + X_i^{1*} = X_i^*$$

$$R(Z) = \left|X_1^{1*}\right|_{1\ 2} + \left|X_2^{1*}\right|_{1\ 1} \left|X_1^{1*}\right|_{\cancel{3}\ \cancel{4}\ 1\ 2} + \left|X_1^{1*}\right|_{1\ 2} \left|X_3^{1*}\right|_{\cancel{5}\ 2} \longrightarrow$$

$$\longrightarrow \left|X_1^{1*}\right|_{1\ 2} + \left|X_2^{1*}\right|_{1\ 1} \left|X_1^{1*}\right|_{1\ 2} + \left|X_1^{1*}\right|_{1\ 2} \left|X_3^{1*}\right|_{2\ 2}$$

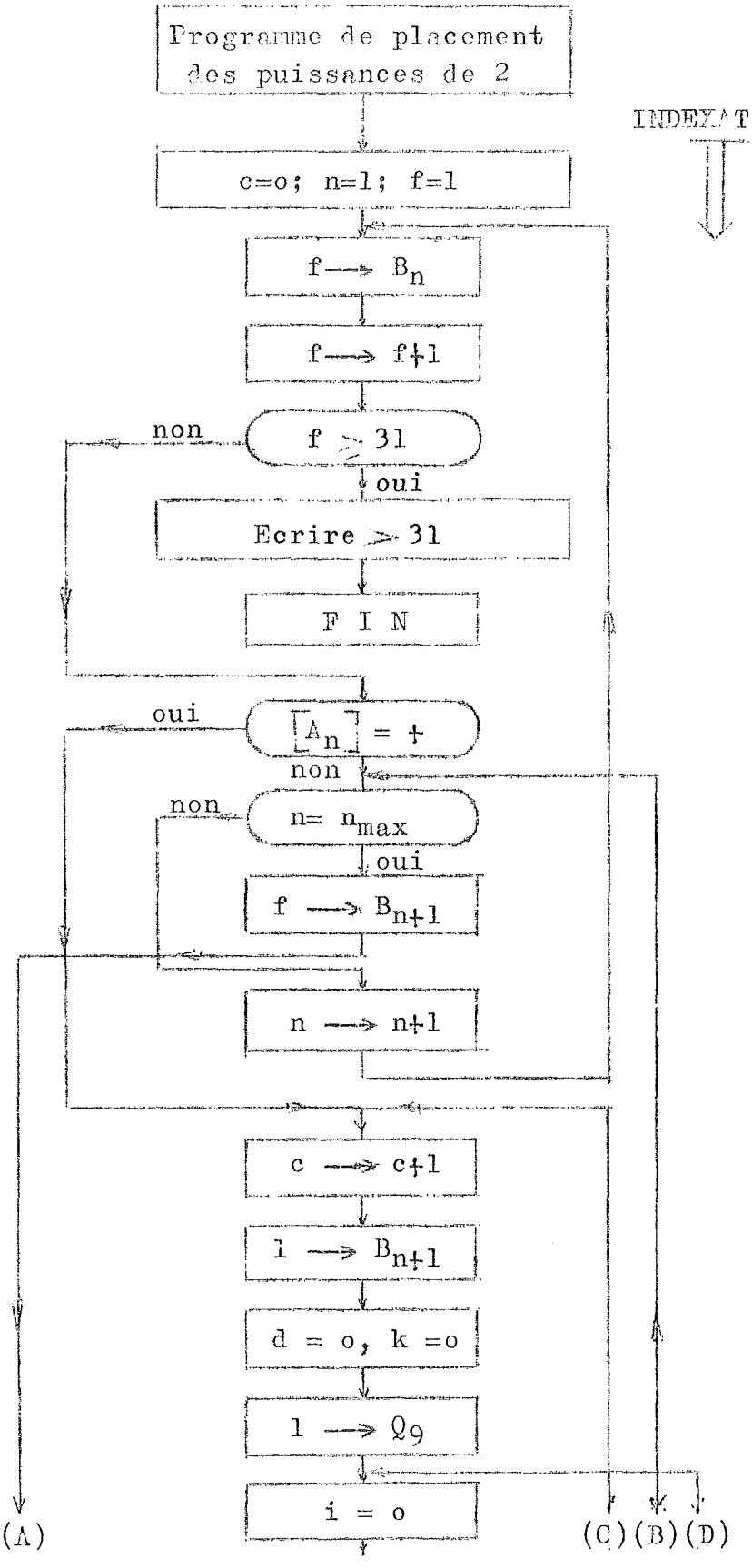
	Z	X_1	X_2	X_3
1	-	1,2	1	1
1,2	Z	1,2	1	1,2

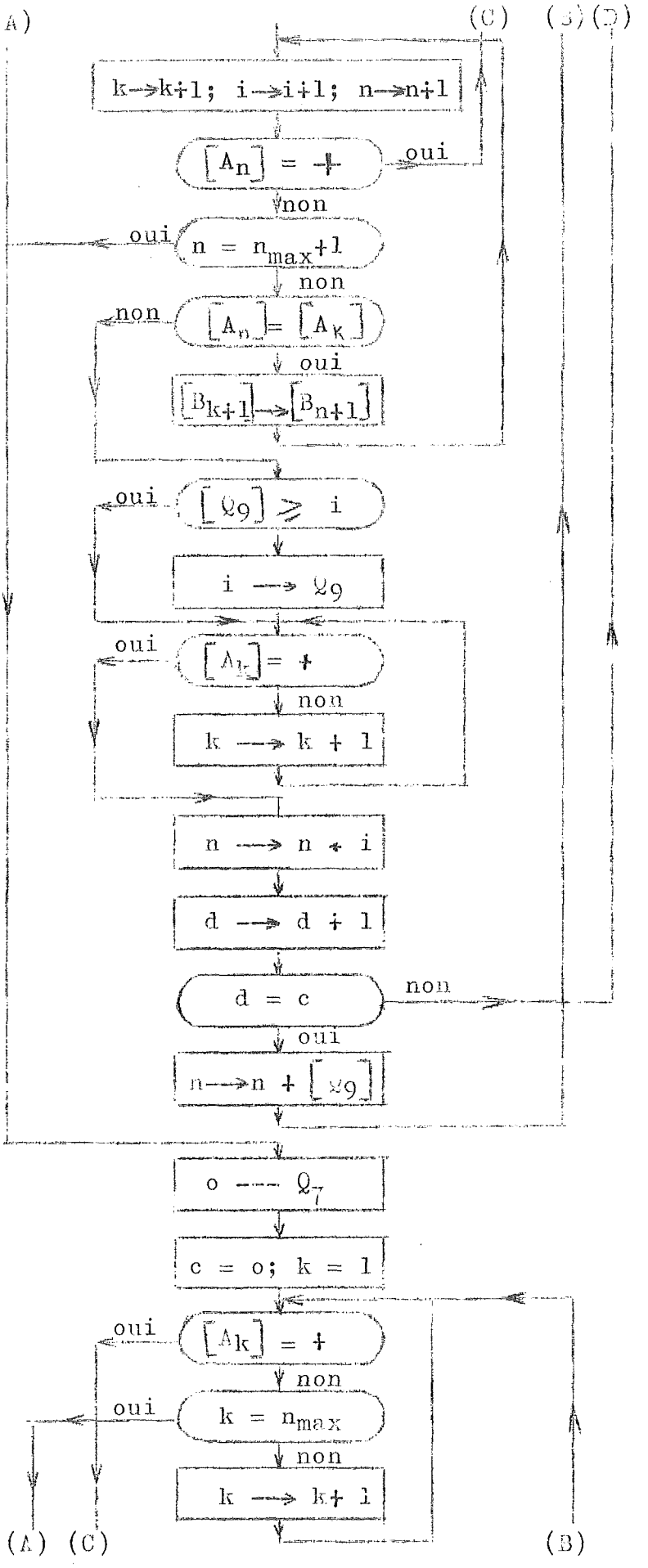
→

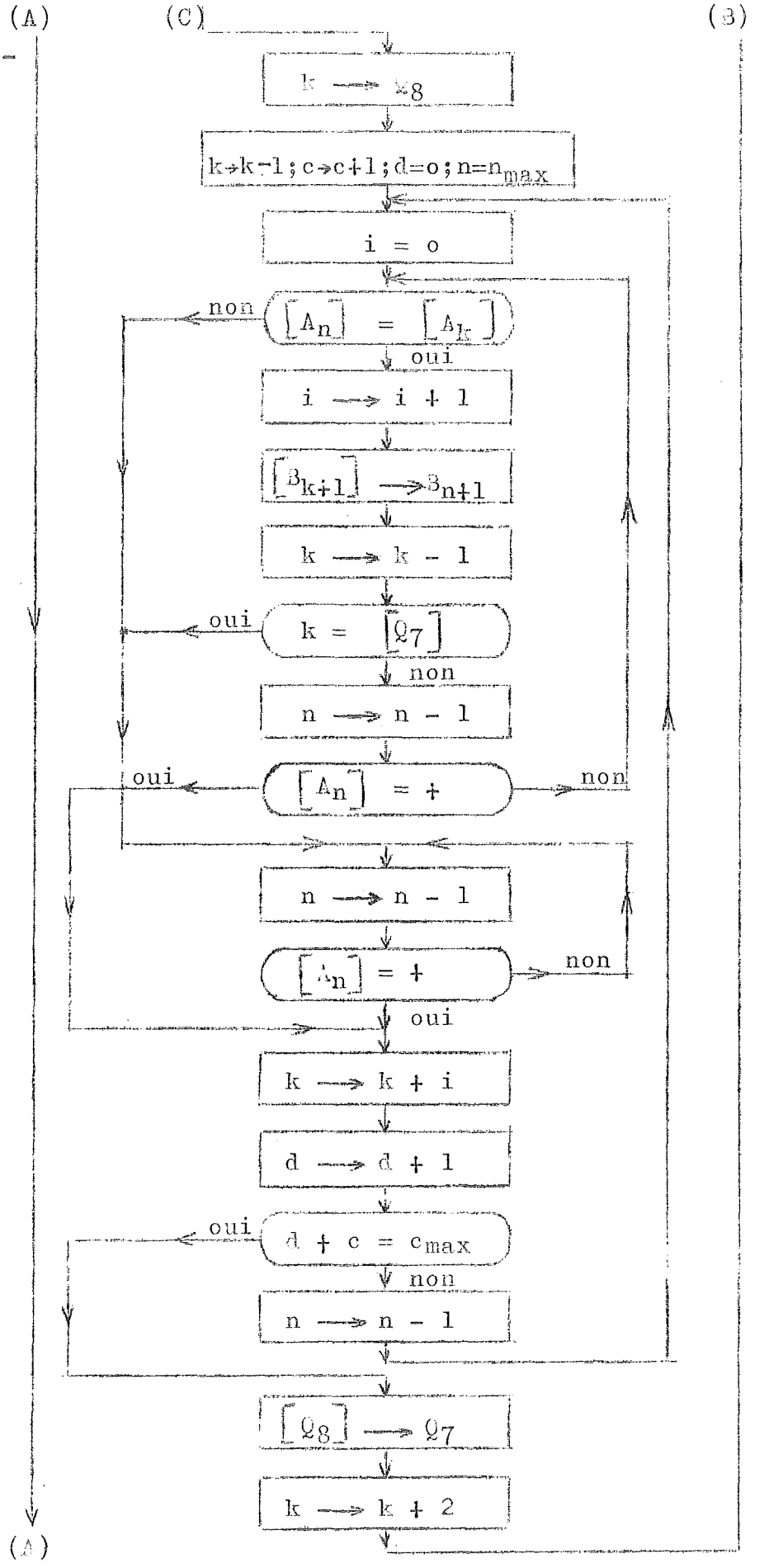
	Z	X_1	X_2	X_3
A	-	B	A	A
B	Z	B	A	B

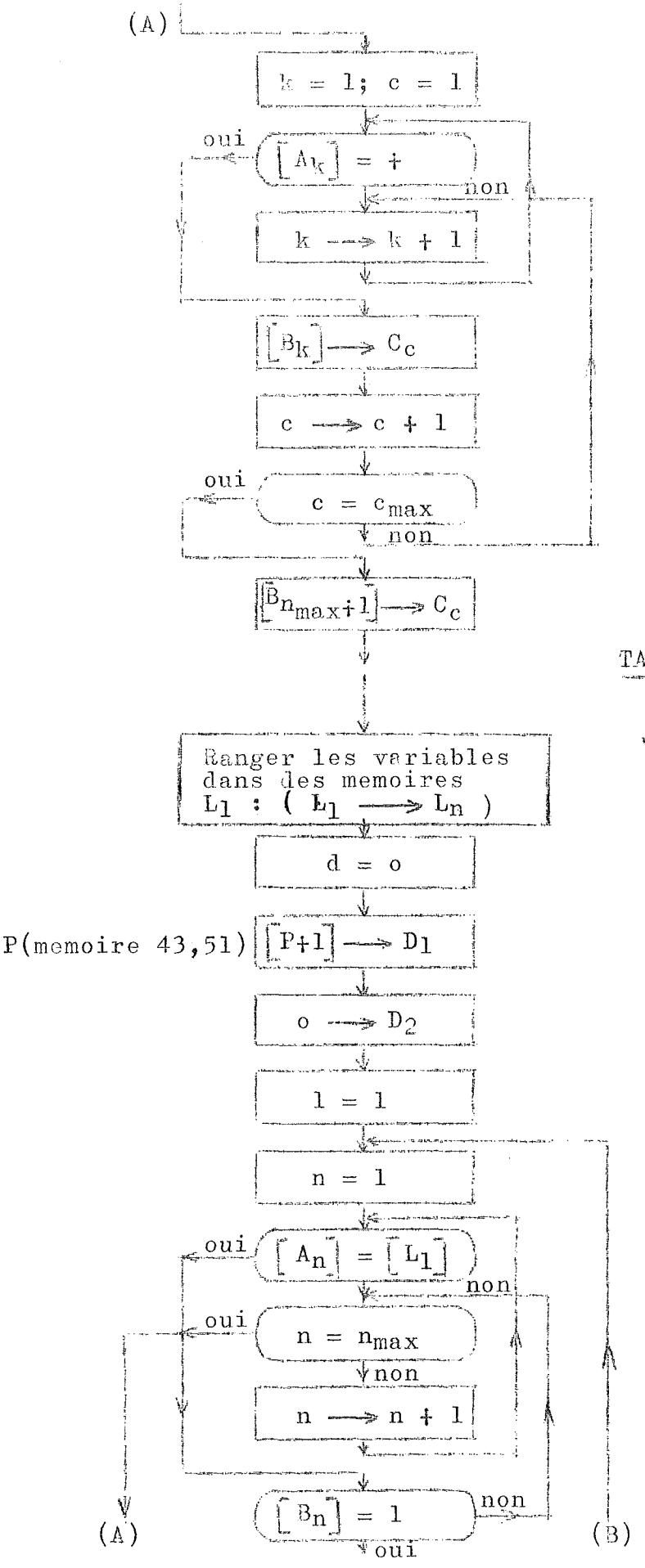
ANNEXE II

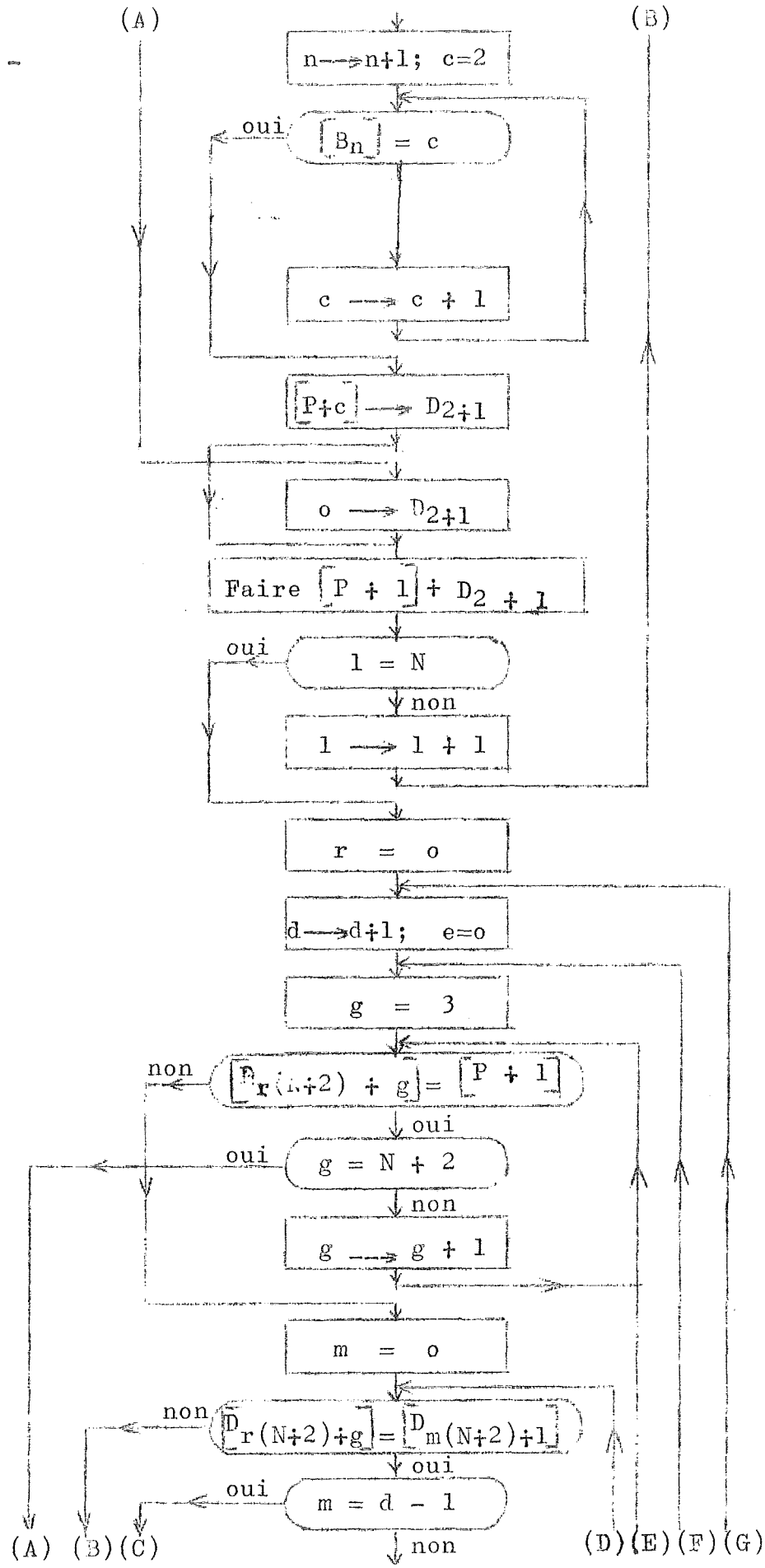
ORGANIGRAMME.-

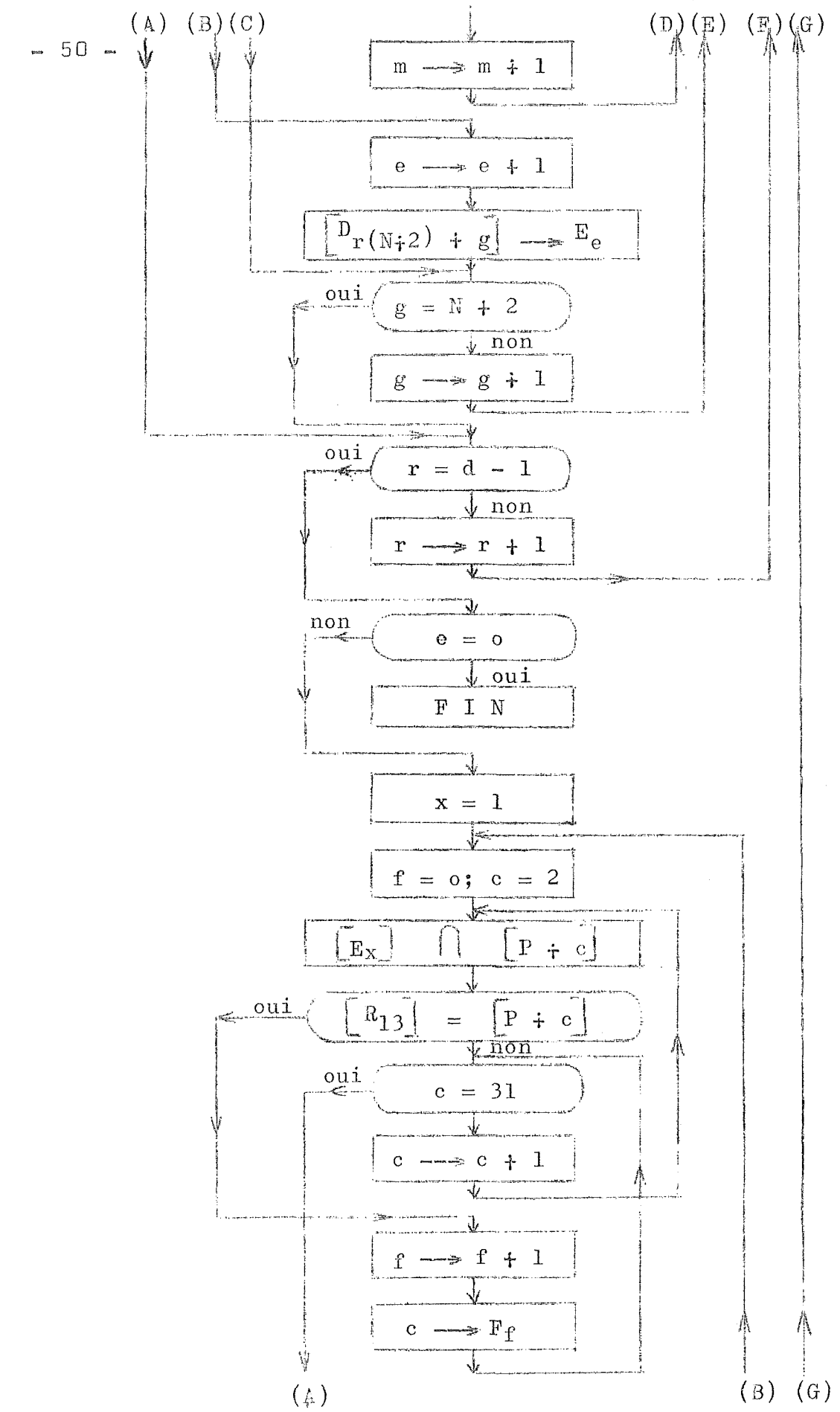


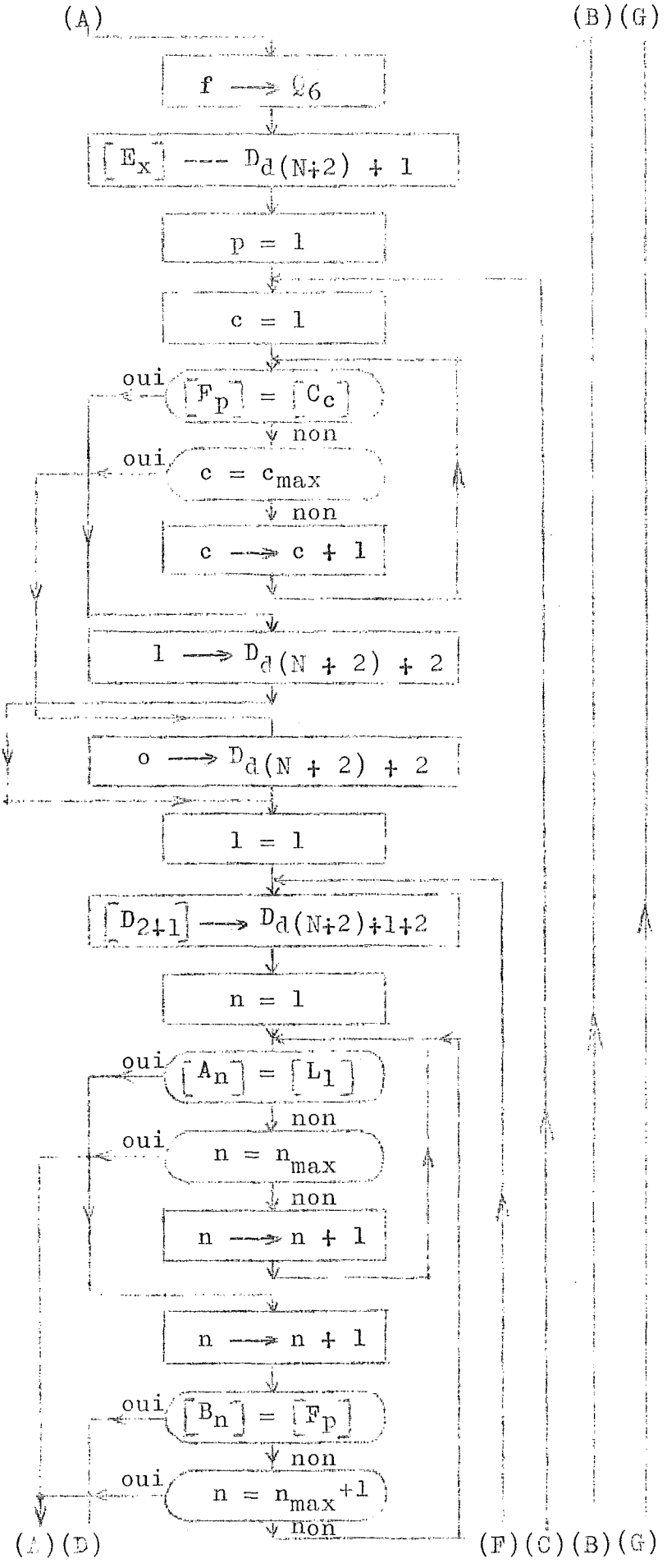

(A)
(C)
(B)
(D)
k→k+1; i→i+1; n→n+1
[An] = +
oui
non
n = nmax+1
oui
non
[An]= [Ak]
non
oui
[Bk+1]→[Bn+1]
[ϱ9] ≥ i
oui
i → ϱ9
[An]= +
oui
non
k → k + 1
n → n + i
d → d + 1
d = c
non
oui
n→n + [ϱ9]
o → Q7
c = o; k = 1
[Ak] = +
oui
non
k = nmax
oui
non
k → k+ 1
(A)
(C)
(B)

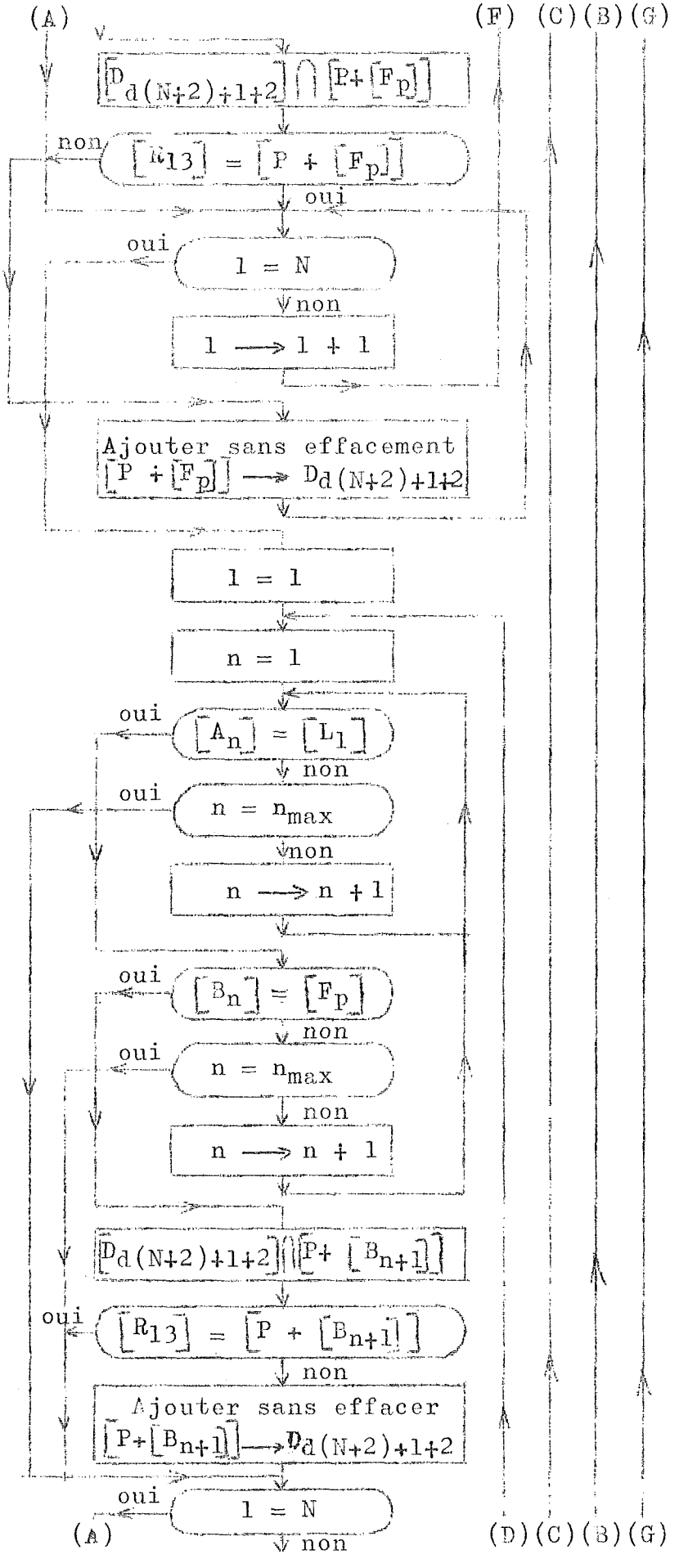

(A)
(C)
(B)
k ⟶ k8
k→k-1 ; c→c+1 ; d=o ; n=n max
i = o
non
[An] = [Ak]
oui
i ⟶ i + 1
[Bk+1] ⟶ Bn+1
k ⟶ k - 1
oui
k = [Q7]
non
n ⟶ n - 1
oui
[An] = +
non
n ⟶ n - 1
[An] = +
non
oui
k ⟶ k + i
d ⟶ d + 1
oui
d + c = cmax
non
n ⟶ n - 1
[Q8] ⟶ Q7
k ⟶ k + 2
(A)

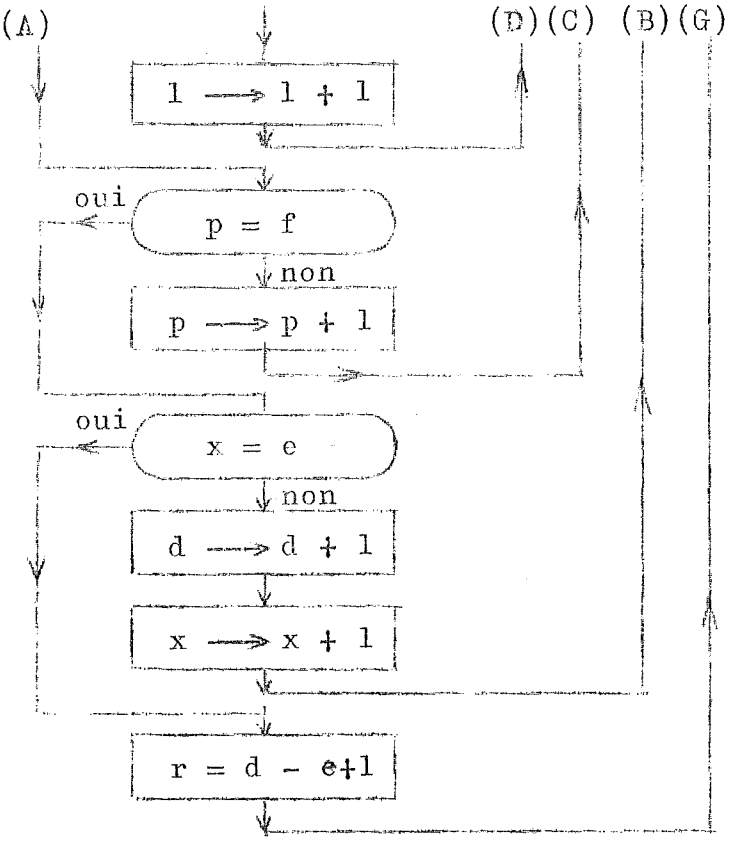

(A)
k = 1; c = 1
oui
[Ak] = +
non
k ⟶ k + 1
[Bk] ⟶ Cc
c ⟶ c + 1
oui
c = cmax
non
[Bnmax+1] ⟶ Cc
TABLE
Ranger les variables dans des memoires
L1 : (L1 ⟶ Ln)
d = o
P(memoire 43,51)
[P+1] ⟶ D1
o ⟶ D2
1 = 1
n = 1
oui
[An] = [L1]
non
oui
n = nmax
non
n ⟶ n + 1
non
[Bn] = 1
oui
(A)
(B)


(A)
(B)
n ⟶ n+1; c=2
oui
[Bn] = c
c ⟶ c + 1
[P+c] ⟶ D2+1
o ⟶ D2+1
Faire [P + 1] + D2 + 1
oui
l = N
non
l ⟶ l + 1
r = o
d ⟶ d+1; e=o
g = 3
non
[Dr(l+2) + g] = [P + 1]
oui
oui
g = N + 2
non
g ⟶ g + 1
m = o
non
[Dr(N+2)+g] = [Dm(N+2)+1]
oui
oui
m = d - 1
non
(A) (B)(C)
(D)(E)(F)(G)


(A) (B) (C)
(D) (E) (F) (G)
m ⟶ m + 1
e ⟶ e + 1
[D r(N+2) + g] ⟶ E e
oui
g = N + 2
non
g ⟶ g + 1
oui
r = d - 1
non
r ⟶ r + 1
non
e = o
oui
F I N
x = 1
f = o; c = 2
[Ex] ∩ [P + c]
oui
[R 13] = [P + c]
non
oui
c = 31
c ⟶ c + 1
f ⟶ f + 1
c ⟶ Ff
(A)
(B) (G)


(A)
(B)(G)
f ⟶ $\mathcal{C}_6$
$[E_x]$ --- $D_{d(N+2)+1}$
p = 1
c = 1
oui
$[F_p] = [C_c]$
non
oui
c = c_{max}
non
c ⟶ c + 1
1 ⟶ $D_{d(N+2)+2}$
o ⟶ $D_{d(N+2)+2}$
l = 1
$[D_{2+1}]$ ⟶ $D_{d(N+2)+1+2}$
n = 1
oui
$[A_n] = [L_1]$
non
oui
n = n_{max}
non
n ⟶ n + 1
n ⟶ n + 1
oui
$[B_n] = [F_p]$
non
oui
n = $n_{max}+1$
non
(A)(D)
(F)(C)(B)(G)


(A)
(F) (C) (B) (G)
$[D_{d(N+2)+1+2}] \cap [P+[F_p]]$
non
$[R_{13}] = [P + [F_p]]$
oui
oui
l = N
non
l ⟶ l + 1
Ajouter sans effacement
$[P + [F_p]] \longrightarrow D_{d(N+2)+1+2}$
l = 1
n = 1
oui
$[A_n] = [L_1]$
non
oui
$n = n_{max}$
non
n ⟶ n + 1
oui
$[B_n] = [F_p]$
non
oui
$n = n_{max}$
non
n ⟶ n + 1
$[D_{d(N+2)+1+2}] \cap [P+[B_{n+1}]]$
oui
$[R_{13}] = [P + [B_{n+1}]]$
non
Ajouter sans effacer
$[P+[B_{n+1}]] \longrightarrow D_{d(N+2)+1+2}$
oui
l = N
(A)
non
(D) (C) (B) (G)


(A)
(D)(C)
(B)(G)
l ⟶ l + 1
oui
p = f
non
p ⟶ p + 1
oui
x = e
non
d ⟶ d + 1
x ⟶ x + 1
r = d - e+1

PROGRAMME.-

```
43,115M
A1EV1
A2EV1
A15ER2
T1M43,50+
S2V31
RV44,0Z2
A2V32
A3ER1
A1R3
RV43,117
X
X
X
A1EV0
A2EV1
A3EV1
A4EV10
A5EM52,111
A15EV0
T2R15
T3M50,0+
A3V1
A9EV31
S9R3
RV44,20P9
X
EV25
EV23
EV52
EV24
EV3
EV1
F
V4TM48,0+
RV44,32Z13
X
A9ER2
S9R5
RV44,28P9
A2V1
RV44,5
A15ER2
A15V1
T3M50,0+
RV44,80
A1V1
A9EV1
A15ER2
A15V1
T9M50,0+
RV45,54
A7EV0
A8EV0
A7V1
A8V1
```

A2V1
A15ER2
V4TM48,0+
RV44,32Z13
X
A9ER2
A10ER5
A10V1
S9R10
RV44,80P9
X
A9EM48,0+
A15ER7
V9TM48,0+
RV45,57Y13
X
A15V1
A9EM50,0+
A15EV1
A15R2
T9M50,0+
RV44,40
A15ER7
V4TM48,0+
RV45,62Z13
X
A7V1
RV44,64
A6V1
V6TR1
RV44,75Y13
X
RV45,64
X
RV44,39
X
X
X
A1EV0
T1M52,117
A7EV1
A15ER7
V4TM48,0+
RV44,93Z13
X
A9ER7
S9R5
RV45,12P9
X
A7V1
RV44,83
T7M52,118
S7V1
A1V1
A6EV0
A2ER5
A8EV0
A15ER2
A9EM48,0+
A15ER7
V9TM48,0+
RV44,122Y13
X
A8V1
A15V1
A9EM50,0+
A15EV1
A15R2
T9M50,0+
S7V1

A9EM52,117
S9R7
RV44,122P9
X
S2V1
A15ER2
V4TM48,0+
RV44,127Z13
X
RV44,99
S2V1
A15ER2
V4TM48,0+
RV44,122Y13
X
A7R8
A6V1
A9ER6
A9R1
S9M52,110
RV45,8P9
X
S2V1
RV44,98
A9EM52,118
T9M52,117
A7V2
RV44,83
A7EV1
A1EV1
A15ER7
V4TM48,0+
RV45,20Z13
X
A7V1
RV45,14
A15ER7
A9EM50,0+
A15ER1
T9M52,0+
A1V1
A9ER1
S9M52,110
RV45,29P9
RV45,18
A15ER5
A15V1
A9EM50,0+
A15ER1
T9M52,0+
X
X
X
A2EV1
A3EV33
A15ER2
T3M52,80+
A9ER2
S9M52,112
RV10,0P9
X
A2V1
A3V1
RV45,39
X
X
X
X
X
X

```
T9M52,119          T8M52,119
A6EVO              RV44,64
RV44,38            S2R8
A9EM52,119         RV44,70
S9R8               A2M52,119
RV44,64P9          RV44,23
```

Exemples traités à la machine

Exemple I.-

$$R(Z) = ACDE + ACD + DB + DCADB$$

Résultats:

```
E50,1D             E52,1D
50,1M 1            52,1M 5
50,2M 2            52,2M 4
50,3M 3            52,3M 7
50,4M 4            52,4M 7
50,5M 5
50,6M 1
50,7M 2            Donc les indices de tous les
50,8M 3            endroits sont les suivants:
50,9M 4            1,2,3,4,5,1,2,3,4,1,6,7,1,6,8,9,
50,10M1 E50,11D    6,7.
50,11M 6           et les endroits finals:
50,12M 7           5,4,7,7.
50,13M 1
50,14M 6
50,15M 8
50,16M 9
50,17M 6
50,18M 7
```

Exemple II.-

R(Z) = BDFGC + ABGE + BD + ABGCBD

Résultats:

E50,1D
50,1M 1
50,2M 2
50,3M 3
50,4M 4
50,5M 5
50,6M 6
50,7M 1
50,8M 7
50,9M 8
5o,10M 5
E50,11D
50,11M 6
50,12M 1
50,13M 2
50,14M 3
50,15M 1
50,16M 7
50,17M 8
50,18M 9
50,19M 10
50,20M 2
E50,21M 3

E52,1D
52,1M 6
52,2M 6
52,3M 3
52,4M 3

Indices de tous les endroits:

1,2,3,4,5,6,1,7,8,5,6,1,2,3,1,
7,8,9,10,2,3.

Endroits finals:

6,6,3,3.

On a verifié aussi avec d'autres exemples le bon fonctionnement du programme.

Nota.- Nous ne donnons pas le texte de la partie table du

programme car il n'a pas fonctionné complètement à cause pensons-nous, d'un faux test qui était situé dans la troisième case (pag. 49) de l'organigramme. Par des raisons differentes on en a abandonnés les essais.

A N N E X E III

PROGRAMME EN LANGAGE FORTRAN IV

On a finalement écrit le programme en langage Fortran IV. Ceci apporte, par rapport au programme précedent, les modifications ci-dessous.

A.11.- Avantages

- Il est utilisable pour n'importe quelle machine à calculer qui posséde un programme compilateur Fortran.

- Il ne présente en soi aucune limitation quant à la taille du problème à traiter. Seule la capacité de la machine est limitative.

- Il a aussi, bien sûr, l'avantage inherent à tout langage symbolique, à savoir qu'il est bien plus compréhensible s'il trouve un lecteur intéressé. D'une façon delibérée nous avons choisi une notation des variables très lourde, mais nous le croyons, très parlante.

A.12.- Inconvenient

- Il faut réserver à l'avance le nombre des mémoires (instruction DIMENSION) dont beaucoup peuvent rester inutilisées. Mais comme les cartes perforées sont permutables, l'on peut toujours choisir un nombre adapté au problème en cours.

A.2.- Données

NVECT = nombre des vecteurs de l'alphabet d'entrée
NPAQU = nombre des paquets de séquences
MAXSEQ = nombre des vecteurs de la séquence la plus longue
NMAXI = nombre des symboles de l'expression régulière
IPLUS = le symbole plus (+++, pour une question de format)
LDOWN(I) , I = 1,NMAXI = l'expression régulière de notre problème
LETTRE(L) , L = 1,NVECT = les vecteurs ou lettres de l'alphabet d'entrée (X01, X02,...)

A.3.- Symboles utilisés dans la partie indexation

La partie indexation du programme Fortran suit fidelement la partie homologue du programme en langage machine, en faisant les transcriptions suivantes des noms des variables:

NMAXI = n_{max}

NVECT = N

NPAQU = c_{max}

LETTRE(L) = L_1

LDOWN(I) = A_n

INDEX(I) = B_n

INDEXF(IPAQU1) = C_c

IPAQU1 = c

IPAQU2 = d

ISYMB1 = k

ISYMB2 = i

INDICE = f

MEMOI9 = Q_9

MEMOI8 = Q_8

MEMOI7 = Q_7

A.4.- Symboles utilisés dans la partie table

Cette partie est bâtie dans le même esprit que son homologue du programme CAB-500 mais diffère assez dans les détails.

Les correspondances avec les symboles de l'organigramme de l'annexe II sont les suivantes:

LGNRT1 ------- r

LGNRT2 ------- m

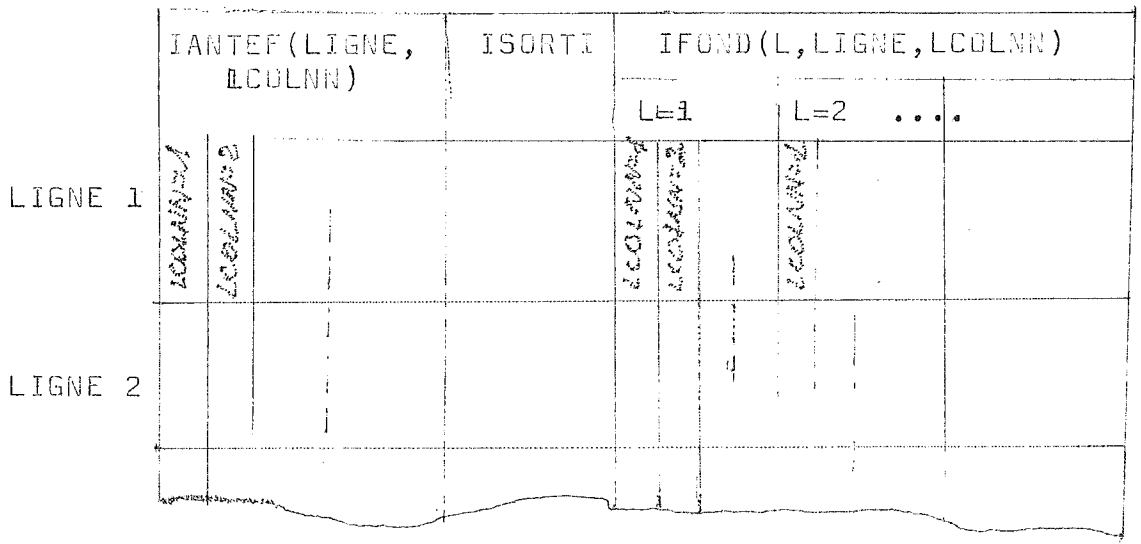
K ------- e

IK ------- x

IANTEF(K, KCOLNN) -- E_e

IANTEF(IK, KCOLNN) -- E_x

On obtient une table par classes d'indices comme suit;



IANTEF(LIGNE,LCOLNN) = classes d'indices des endroits antéfondamentaux

ISORTI(LIGNE) = sortie

IFOND(L,LIGNE,LCOLNN) = classes d'indices des endroits fondamentaux

A partir de cette table on obtient celle codifiée par états, avec cet arrangement:

LANTEF(LIGNE) = états actuels

ISORTI(LIGNE) =sortie

IFOND(L,LIGNE,1) = états suivants (on réutilise ces mémoires par un souci d'épargne)

Les mémoires M(MIN) servent à arranger par ordre croissant les indices d'une classe antéfondamentale.

A.5.- Programme

```
C       PROGRAMME INDEXATION EXPRESSIONS REGULIERES
        DIMENSION LANTEF(40)
        DIMENSION LDONN(40),LETTRE(5),INDEX(40),INDEXF(5),
            IANTEF(40,5),ISORTI(40),IFOND(5,40,=),M(5)
        READ(5,1001)NVECT,NPAQU,MAXSEQ,NMAXI,IPLUS
 1001   FORMAT(3I2,I5,A3)
        READ(5,1002)(LDONN(I),I=1,NMAXI)
 1002   FORMAT(24A3)
        READ(5,1003)(LETTRE(L),L=1,NVECT)
 1003   FORMAT(24A3)
        IPAQU1=0
        I=1
        INDICE=1
 1      INDEX(I)=INDICE
        INDICE=INDICE+1
        IF(LDONN(I)-IPLUS)7,10,7
 7      IF(NMAXI-I)8,8,6
 6      I=I+1
        GO TO 1
 8      INDEX(I+1)=INDICE
        GO TO 24
 10     IPAQU1=IPAQU1+1
        INDEX(I+1)=1
        IPAQU2=0
        ISYMB1=0
        MEMOI9=1
 15     ISYMB2=0
 16     ISYMB1=ISYMB1+1
        ISYMB2=ISYMB2+1
        I=I+1
        IF(LDONN(I)-IPLUS)11,10,11
```

```
11    IF(I-NMAXI-1)17,24,24
17    IF(LDONN(I)-LDONN(ISYMB1))19,18,19
18    INDEX(I+1)=INDEX(ISYMB1+1)
      GO TO 16
19    IF(MEMOI9-ISYMB2)20,21,21
20    MEMOI9=ISYMB2
21    IF(LDONN(ISYMB1)-IPLUS)25,22,25
25    ISYMB1=ISYMB1+1
      GO TO 21
22    I=I-ISYMB2
      IPAQU2=IPAQU2+1
      IF(IPAQU2-IPAQU1)15,23,23
23    I=I+MEMOI9
      GO TO 7
24    MEMOI7=0
      IPAQU1=0
      ISYMB1=1
31    IF(LDONN(ISYMB1)-IPLUS)27,35,27
27    IF(ISYMB1-NMAXI)33,50,50
33    ISYMB1=ISYMB1+1
      GO TO 31
35    MEMOI8=ISYMB1
      ISYMB1=ISYMB1-1
      IPAQU1=IPAQU1+1
      IPAQU2=0
      I=NMAXI
36    ISYMB2=0
37    IF(LDONN(I)-LDONN(ISYMB1))42,38,42
38    ISYMB2=ISYMB2+1
      INDEX(I+1)=INDEX(ISYMB1+1)
      ISYMB1=ISYMB1-1
      IF(ISYMB1-MEMOI7)42,42,39
39    I=I-1
      IF(LDONN(I)-IPLUS)37,43,37
```

```
42      I=I-1
        IF(LDONN(I)-IPLUS)42,43,42
43      ISYMB1=ISYMB1+ISYMB2
        IPAQU2=IPAQU2+1
        IF(IPAQU2+IPAQU1-NPAQU)44,45,46
44      I=I-1
        GO TO 36
45      MEMOI7=MEMOI8
        ISYMB1=ISYMB1+2
        GO TO 31
50      ISYMB1=1
        IPAQU1=1
51      IF(LDONN(ISYMB1)-IPLUS)52,53,52
52      ISYMB1=ISYMB1+1
        GO TO 51
53      INDEXF(IPAQU1)=INDEX(ISYMB1)
        IPAQU1=IPAQU1+1
        IF(IPAQU1-NPAQU)52,54,54
54      INDEXF(IPAQU1)=INDEX(NMAXI+1)
1004    FORMAT('  INDEXATION'//6(2X,A3,2X,I2)//)
        WRITE(6,1004)(LDONN(I),INDEX(I+1),I=1,NMAXI)
        IANTEF(1,1)=1
        DO 56 L=1,NVECT
56      IFOND(L,1,1)=1
        MMM=NMAXI+1
        DO 57 LIGNE=1,MMM
        ISORTI(LIGNE)=0
        DO 57 LCOLNN=2,MAXSEQ
        IANTEF(LIGNE,LCOLNN)=0
        DO 57 L=1,NVECT
57      IFOND(L,LIGNE,LCOLNN)=0
        LIGNE=1
        DO 78 L=1,NVECT
60      I=1
61      IF(LDONN(I)-LETTRE(L))62,64,62
```

```
62      IF(I-NMAXI)63,78,78
63      I=I+1
        GO TO 61
64      IF(INDEX(I)-1)62,65,62
65      I=I+1
        IFOND(L,LIGNE,2)=INDEX(I)
78      CONTINUE
        LGNRT1=1
85      K=LIGNE
        LIGNE=LIGNE+1
        MEMOI6=LIGNE
86      L=1
87      IF(IFOND(L,LGNRT1,2))90,88,90
88      IF(L-NVECT)89,100,100
89      L=L+1
        GO TO 87
90      LGNRT2=1
91      DO 92 LCOLNN=2,MAXSEQ
        IF(IFOND(L,LGNRT1,LCOLNN)-IANTEF(LGNRT2,LCOLNN))95,
                                                        92,95
92      CONTINUE
        GO TO 97
95      IF(LGNRT2-LIGNE+1)94,93,93
94      LGNRT2=LGNRT2+1
        GO TO 91
93      K=K+1
        DO 96 LCOLNN=1,MAXSEQ
        KCOLNN=LCOLNN
96      IANTEF(K,KCOLNN)=IFOND(L,LGNRT1,LCOLNN)
97      IF(L-NVECT)99,100,100
99      L=L+1
        GO TO 87
100     IF(LGNRT1-LIGNE+1)101,102,102
101     LGNRT1=LGNRT1+1
```

```
      GO TO 86
102   IF(K-LIGNE+1)120,103,120
103   WRITE(6,1006)(LIGNE?(IANTEF(LIGNE,LCOLNN),LCOLNN=1,
            MAXSEQ),ISORTI(LIGNE),((IFOND(L,LIGNE,LCOLNN),
            LCOLNN=1,MAXSEQ),L=1,NVECT),LIGNE=1,INDICE)
1006  FORMAT(//'  TABLE PAR CLASSES D,INDICES'//20(2X,I2)//)
      DO 73 LIGNE=1,INDICE
      IF(IANTEF(LIGNE,1))70,71,70
70    LANTEF(LIGNE)=LIGNE
      GO TO 73
71    LANTEF(LIGNE)=0
73    CONTINUE
      DO 72 JM=1,INDICE
      DO 72 LIGNE=1,INDICE
      DO 72 L=1,NVECT
      DO 74 LCOLNN=1,MAXSEQ
      IF(IANTEF(JM,LCOLNN)-IFOND(L,LIGNE,LCOLNN))72,74,72
74    CONTINUE
      IFOND(L,LIGNE,1)=LANTEF(JM)
72    CONTINUE
      WRITE(6,1008)(LANTEF(LIGNE),ISORTI(LIGNE),(IFOND(L,
            LIGNE,1),L=1,NVECT),LIGNE=1,INDICE)
1008  FORMAT(//'  TABLE PAR ETATS'//7(3X,I2)//)
      STOP
120   IK=LIGNE
121   DO 123 L=1,NVECT
      DO 123 LCOLNN=1,MAXSEQ
123   IFOND(L,LIGNE,LCOLNN)=IFOND(L,1,LCOLNN)
      KCOLNN=2
124   IF(IANTEF(IK,KCOLNN))125,167,125
125   DO 126 IPAQU1=1,NPAQU
      IF(IANTEF(IK,KCOLNN)-INDEXF(IPAQU1))126,128,126
126   CONTINUE
      GO TO 130
```

```
128   ISORTI(LIGNE)=1
130   L=1
122   DO 137 LCOLNN=1,MAXSEQ
      IF(IANTEF(IK,KCOLNN)-IFOND(L,LIGNE,LCOLNN))137,150,137
137   CONTINUE
127   I=1
131   IF(LDONN(I)-LETTRE(L))132,135,132
132   IF(I-NMAXI)133,150,150
133   I=I+1
      GO TO 131
135   I=I+1
      IF(INDEX(I)-IANTEF(IK,KCOLNN))136,138,136
136   IF(I-NMAXI-1)131,150,150
138   DO 140 LCOLNN=2,MAXSEQ
      IF(IFOND(L,LIGNE,LCOLNN))140,141,140
140   CONTINUE
141   IFOND(L,LIGNE,LCOLNN)=INDEX(I)
150   IF(L-NVECT)151,152,152
151   L=L+1
      GO TO 122
152   L=1
153   I=1
154   IF(LDONN(I)-LETTRE(L))155,157,155
155   IF(I-NMAXI)156,164,164
156   I=I+1
      GO TO 154
157   IF(INDEX(I)-IANTEF(IK,KCOLNN))158,160,158
158   IF(I-NMAXI)159,164,164
159   I=I+1
      GO TO 154
160   DO 162 LCOLNN=2,MAXSEQ
      IF(IFOND(L,LIGNE,LCOLNN)-INDEX(I+1))161,164,161
161   IF(IFOND(L,LIGNE,LCOLNN))162,163,162
162   CONTINUE
```

```
163   IFOND(L,LIGNE,LCOLNN)=INDEX(I+1)
164   IF(L-NVECT)165,166,166
165   L=L+1
      GO TO 153
166   KCOLNN=KCOLNN+1
      GO TO 124
167   DO 180 L=1,NVECT
      IPETIT=IFOND(L,LIGNE,2)
      MIN=1
      M(MIN)=1
174   DO 168 LCOLNN=2,MAXSEQ
      IF(IFOND(L,LIGNE,LCOLNN))175,168,175
175   IF(IFOND(L,LIGNE,LCOLNN)-IPETIT)169,169,168
169   IPETIT=IFOND(L,LIGNE,LCOLNN)
      ICOLNN=LCOLNN
168   CONTINUE
      IFOND(L,LIGNE,ICOLNN)=0
      MIN=MIN+1
      M(MIN)=IPETIT
      IF(M(MIN-1)-M(MIN))171,170,170
171   GO TO 174
170   M(MIN)=0
      DO 176 LCOLNN=2,MIN
176   IFOND(L,LIGNE,LCOLNN)=M(LCOLNN)
      J=MIN+1
      DO 180 LCOLNN=J,MAXSEQ
180   IFOND(L,LIGNE,LCOLNN)=0
181   IF(IK-K)182,183,183
182   LIGNE=LIGNE+1
      IK=IK+1
      GO TO 121
183   LGNRT1=MEMOI6
      GO TO 85
      END
```